НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“ ” 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки ***6.050103 «Програмна інженерія»***

на тему *Програмне забезпечення порівняння електронних та сканованих документів*

Виконав: студент IV курсу, групи *ІП-52 Олійник Аліна Олексіївна* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст.викл., Олійник Ю.О.* _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Рецензент:

_____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – *6.050103 «Програмна інженерія» (Програмне забезпечення систем)*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Олійник Аліни Олексіївни
(прізвище, ім’я, по батькові)

1. Тема проекту «Програмне забезпечення порівняння електронних та сканованих документів»

керівник проекту *Олійник Юрій Олександрович, ст.викл.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “23” квітня 2019 р. №1181-с

2. Термін подання студентом проекту «03» *червня 2019 року*

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1) Аналіз вимог до програмного забезпечення: загальні положення, змістовий опис і аналіз предметної області, математичне забезпечення, опис процесу діяльності огляд існуючих технічних рішень та відомих, аналіз успішний IT-проектів, аналіз вимог до програмного забезпечення.

2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, аналіз безпеки даних.

3) Аналіз якості та тестування програмного забезпечення: аналіз якості ПЗ, опис

процесів тестування, опис контрольного прикладу.

4) Впровадження та супровід програмного забезпечення: розгортання програмного забезпечення, робота з програмним забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна діяльності роботи програмного забезпечення

2) Схема бізнес-процесів роботи програмного забезпечення

3) Схема структурна класів програмного забезпечення

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «20» *лютого 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*22.02.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*22.02.2019*	
3.	*Постановка та формалізація задачі*	*1.03.2019*	
4.	*Аналіз вимог до програмного забезпечення*	*5.03.2019*	
5.	*Алгоритмізація задачі*	*15.03.2019*	
6.	*Моделювання програмного забезпечення*	*20.03.2019*	
7.	*Обґрунтування використовуваних технічних засобів*	*25.03.2019*	
8.	*Розробка архітектури програмного забезпечення*	*05.04.2019*	
9.	*Розробка програмного забезпечення*	*10.04.2019*	
10.	*Налагодження програми*	*15.05.2019*	
11.	*Виконання графічних документів*	*20.05.2019*	
12.	*Оформлення пояснювальної записки*	*20.05.2019*	
13.	*Подання ДП на попередній захист*	*28.05.2019*	
14.	*Подання ДП рецензенту*	*28.05.2019*	
15.	*Подання ДП на основний захист*	*07.06.2019*	

Студент _____ Олійник А.О.
(підпис)

Керівник проекту _____ Олійник Ю.О.
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	A4		Завдання на дипломний проект	2	
2	A4	КПІ.ІП-5213.045430.02.81	Пояснювальна записка	51	
3	A4	КПІ.ІП-5213.045430.03.91	Технічне завдання	11	
4	A4	КПІ.ІП-5213.045430.04.51	Програма та методика тестування	6	
5	A4	КПІ.ІП-5213.045430.05.34	Керівництво користувача	9	
6	A4	КПІ.ІП-5213.045430.06.13	Опис програми	11	
7	A3	КПІ.ІП-5213.045430.07.99.СС	Схема структурна діяльності	1	
8	A3	КПІ.ІП-5213.045430.07.99.СБП	Схема бізнес-процесів	1	
9	A3	КПІ.ІП-5213.045430.07.99.СС	Схема структурна класів програмного забезпечення	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-5213.045430.00.90		
Розробн.		Олійник А.О.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Олійник Ю.О.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-52	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка складається з 4 розділів, містить 5 рисунків, 17 таблиць, 3 додатків, 3 графічних схема, 13 посилань. Загальна кількість сторінок – 51.

Дипломний проект присвячений розробці програмного забезпечення порівняння електронних та сканованих документів.

Метою цієї роботи є підвищення ефективності роботи користувача при роботі обробки паперових документів та порівнянні їх з електронною копією версії.

У першому розділи наведені загальні положення, опис та аналіз предметної області, сформульовані функціональні і нефункціональні вимоги, головна мета, цілі та призначення програми.

У другому розділі описано моделювання та бізнес-процеси програмного забезпечення, розроблена архітектура та компоненти. Наведено детальний опис модулів, класів та функцій.

У третьому розділі проаналізовано якість програмного забезпечення, описано процес тестування та наведені приклади тестів.

У четвертому розділі наведено опис розгортання та роботи програми.

Результатом роботи над проектом є програмне забезпечення, що порівнює електронні та скановані документи.

РОЗПІЗНАВАННЯ ТЕКСТУ, ПОРІВНЯННЯ, ЕЛЕКТРОННИЙ ТА СКАНОВАНИЙ ДОКУМЕНТ, ЗОБРАЖЕННЯ.

ABSTRACT

The explanatory note of the diploma project consists of 4 sections, 5 pictures, 17 tables, 3 additions, 3 graphic diagrams, 13 sources. Total number of pages is 51.

The diploma project is devoted to the development of software for comparing electronic and scanned documents.

The aim of the diploma project is to increase the efficiency of the user's work in processing paper documents and compare them with the electronic version of document.

The first chapter provides general provisions, description and analysis of the subject area, formulated functional and non-functional requirements, the main aim, objectives and purpose of the program.

The second chapter describes the modeling and business process of software, architecture and components. The detailed description of modules, classes and functions is given.

The third chapter analyzes the quality of the software, describes the testing process and gives examples of tests.

The fourth chapter describes how to deploy the program and how to work with the program.

The result of the diploma project is software that compares electronic and scanned documents.

TEXT RECOGNITION, COMPARING, ELECTRONCIC AND SCANNED DOCUMENT, IMAGE.

Пояснювальна записка
до дипломного проекту

на тему: Програмне забезпечення порівняння електронних та сканованих документів _____
_____
_____

Київ – 2019 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Відстань Дамерау-Левенштейна – міра відмінності двох послідовних рядків, що обчислюється мінімальною кількістю операцій вставки, видалення, заміни та транспозиції перетворення одного рядка в іншого. Являє собою модифікацію відстані Левенштейна.

Документ – інформація, що зафіксована в електронному або паперовому вигляді з певними реквізитами.

BPMN – система умовних позначень (нотація) для моделювання бізнес-процесів.

DOC, DOCX, DOT, DOTX, RTF – формат електронних документів, які створюються, редагуються та відкриваються за допомогою програми Microsoft Word.

DPI – показник кількості точок на дюйм, використовується визначення розміру зображення.

JPG, JPEG, PNG, TIFF – формати зображень.

JSON (JavaScript Object Notation) – текстовий формат, що дозволяє описувати об’єкти та інші структури даних для обміну інформації.

OCR (Optical Character Recognition) – технологія, що дозволяє електронно перетворити друковані або рукописні символи в машино-кодований текст.

PDF – формат електронних документів, створений компанією Adobe Systems, який можна відкрити під керівництвом будь-якої операційної системи.

ВСТУП

На даному етапі розвитку технологій все частіше використовується електронний документообіг. Таке рішення дає змогу надовго зберігати важливу документацію, надавити доступ до неї багатьом користувачам, а також швидко створювати, редагувати, перевіряти та аналізувати. Але від використання паперових версій документів досі не відмовляються, тому застосування програмних забезпечень, що швидко та легко перетворюють паперовий варіант в електронний, стає все більше актуальним.

Також, при існування проблеми зберігання документів в обох видах, виникає інша проблема ідентифікації, співставлення, порівняння та визначення ідентичності інформації на них. Буває, що дані можуть бути застарілими, зміненими або помилковими. В такому випадку для вирішення проблеми можна використовувати програму, що порівнює відомості документів та виводить потрібний результат про відповідність інформації на обох типах. Також важливим завданням є можливість підкреслення та виводу невідповідних даних для знаходження їх розташування, порівняння та виправлення на нові відомості. Таке рішення полегшувало роботу користувача та зменшувало ймовірність помилки при виконанні визначення ідентичності документів. З цієї причини дана тема для розвитку інформаційних технологій є актуальною.

Сучасні популярні рішення пропонують автоматизований процес вирішення усіх вище наведених проблем, та вони мають ряд недоліків. Прикладами їх мінусів у використання є платна версія продукту або безкоштовна лише на тестовий період, незрозумілий інтерфейс, відсутність можливості розпізнавання тексту на зображеннях, що приводить до неможливості порівнювати графічний формат документу з іншими.

Метою цієї роботи є підвищення ефективності роботи користувача при роботі обробки паперових документів та порівнянні їх з електронною копією версії та зменшенні ймовірності помилок при аналізі та висновках.

Завданням створення даної роботи є програмне забезпечення, що стане безкоштовним для використання та надасть можливість порівнювати документи багатьох форматів, включаючи графічних та відтворювати всю необхідну інформацію для аналізу та виправлення даних.

Практичним застосуванням такої програми є порівняння електронних та сканованих документів при роботі з електронним документообігом.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

На даному етапі розвитку більш актуальним стає зберігання потрібних документів в електронному типі, але від паперового вигляду ніхто не відмовляється. Задача перенесення даних з одного виду на інший є громісткою та вимагає певного часу та уважності з боку людини. Найпростішим та швидким способом обробки паперової версії на електронну є фотографування та сканування документа. В такому разі виникають проблеми з редагуванням, пошуку та аналізу даних. Наразі технології, що можуть розпізнати текст та порівняти його з іншим в будь-якому іншому форматі, стають дуже популярними та затребуваними через зміну документообігу на електронну версію. Тому така задача вже має свої варіанти вирішення, що полегшує роботу довгострокового збереження інформації, підвищує ефективність роботи користувача з документами та надає одночасний доступ до декількох людей. Задачею розпізнавання машинописного тексту являється оптичне розпізнавання символів (optical character recognition, OCR). Перша така система була створена в 1970-х роках, а з того часу її вирішення модифікуються, а результат поліпшується. Сьогодні такі системи з великою точністю можуть видати усі символи, що зображені на папері, на електронному пристрої, незалежно від їх шрифту, розміру або положенні.

1.2 Змістовний опис і аналіз предметної області

OCR (Optical Character Recognition) – технологія, що дозволяє електронно перетворити друковані або рукописні символи в машино-кодований текст [1]. Це поширений метод оцифровки друкованих текстів, для подальшого редагування в електронному вигляді, пошуку, компактнішого зберігання, відображення, аналізу. Системи розпізнавання тексту можна

розділити на 2 категорії: онлайн, що працюють в режимі реального часу, та оффлайн, які обробляють локальні документи.

Можна виділити декілька методів розпізнавання тексту, а саме: шаблонний метод, статистичний метод, структурний метод та нейронні мережі. Їх основна відмінність - це набір характеристик, що ідентифікуються.

Шаблонний підхід використовується тоді, коли потрібно визначити подібність між об’єктами на зображенні та шаблоном. Для знаходження степені схожості використовується мінімум (максимум) функції співставлення зображення та його прототипу. Такий метод просто використовувати, так як потрібно визначити прототип та потрібну функцію З іншої сторони він не дає точно результату, якщо є незначне відхилення від шаблону. Приклад використання шаблонного методу представлений на рисунку 1.1. На частині рисунку (а) зображене співпадіння з шаблоном, (б) вказує на відхилення, (в) – зміна розміру символу, (г) показує поворот тексту на декілька градусів.

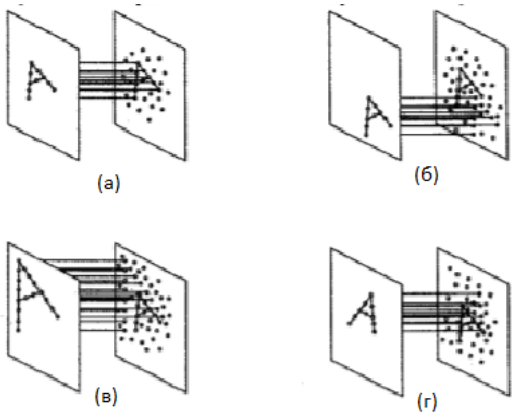


Рисунок 1.1 – Опис використання шаблонного методу [2]

Для використання статичного методу необхідно визначити статичні функції прийняття рішень та критерій оптимальності. Друга складова розраховує показник ймовірності виникнення закономірностей між зображенням та шаблоном певного класу [3].

Структурний метод - це рекурсивний опис складного об’єкту за допомогою примітивів (простих шаблонів). Саме зображення представляється у вигляді зв’язків між ними. Простим прикладом примітивів є алфавіт, тому зображення текстового документу можна представити у вигляді простих

шаблонів та граматики [4]. Перевагою такого методу є не важливе значення розміру букв та шрифтів.

Останнім методом є нейронні мережі. Нейронними мережами можна назвати масивні паралельно обчислювальні системи, які мають велику кількість нейронів, що взаємодіють між собою. Для визначення готовності мережі використовують функцію втрат. При її мінімальному значенні якість роботи вважається високою [5]. Перевагами даного методу є те, що він, порівняно з іншими, видає досить точний результат, при цьому не враховуючи розмір, шрифт або положення символів. З іншої сторони розробка алгоритму вимагає теоретичні знання для аналізу математичної моделі.

Основні підходи розпізнавання тексту, їх переваги та недоліки наведені в таблиці 1.1.

Таблиця 1.1 – Порівняння методів розпізнавання

Назва підходу	Переваги	Недоліки
Шаблонний метод	Простий для використання та розробки алгоритму.	Існує строга залежність від шаблону, будь-яке відхилення дасть помилкові результати; є ймовірність виведення помилок.
Статистичний метод	Видає оцінку результату; є ефективним при використання на обмеженій кількості даних для навчання.	При великих кількостях даних повільний, існує ймовірність виводу неточних даних.
Структурний підхід	Не залежить від розміру, шрифту та положення символів.	Якщо містить багато примітивів, то складно визначити клас зображення.

Продовження таблиці 1.1.

Нейронні мережі	Видають точний результат за малий проміжок часу.	Складний через наявності при розробці теоретичної та математичної основи.

Отже, в результаті порівняння наведених методів можна визначити, що для простого та зрозумілого використання кращий є шаблонний метод, та він не завжди видає точні результати. З іншої сторони на сьогодні популярнішим стає підхід нейронних мереж, так як він видає точні результати та працює швидше [6].

Важливо пам'ятати, що будь-яка система оптичного розпізнавання не ідеальна, і можливо, що текст буде розпізнаний не точно. Незважаючи на це, ця технологія наразі постійно вдосконалюється та налаштовується для забезпечення більшої точності.

1.3 Математичне забезпечення

Алгоритм розрахунку відстані Дамерау-Левенштейна використовується для визначення показника відмінності двох послідовних рядків. Міра обчислюється як мінімальна кількість операцій вставки, видалення, заміни та транспозиції перетворення одного рядка в іншого. Цей алгоритм являє собою модифікацію відстані Левенштейна.

Щоб виразити відстань між двома строками a і b, використовується функція d, яка визначає відстань між i-тим символом a та j-тим символом b. Знаходиться функція за такою формулою:

$$d_{a,b}(i,j)=\min\begin{cases}0 & if\ i=j=0\\ d_{a,b}(i-1,j)+1 & if\ i>0\\ d_{a,b}(i,j-1)+1 & if\ j>0\\ d_{a,b}(i-1,j-1)+1_{(a\neq b)} & if\ i,j>0\\ d_{a,b}(i-2,j-2)+1 & if\ i,j>0, a[i]=b[j-1]\ and\ a[i-1]=b[j]\end{cases} \quad (1)$$

, де другий рядок означає видалення, третій вставлення, четвертий визначає чи співпадають символи, останній означає транспозицію [7].

Порівняно з першою версією алгоритму, даний має перевагу у розрахунку та визначенні результату через додатковий параметр транспозицію.

Алгоритм працює за час *O(m • n)*, де *m* довжина строки *a*, а *n* – довжина строки *b.*Тому при великих об’ємах символів функція буде працювати довго та витратить багато пам’яті.

1.4 Опис процесу діяльності.

Першим етапом роботи процесу є завантажування потрібного документу(ів). Важливо пам’ятати, що зображення має бути певного розміру та чим більш якісне воно, тим точніше результати розпізнавання тексту.

Після цього виконується безпосередньо розпізнавання символів. На даному етапі впроваджуються готові технологічні рішення, використовуються бібліотеки при створенні програми або власноруч розробляється алгоритм з вибраним підходом.

Далі визначається тип документу за допомогою співставлення з шаблонами. Наприклад, до класів документів можна віднести заяву, статтю, чек і так далі.

Одночасно з цим виконується розпізнавання або зчитування тексту. Дані можна поділити на 2 типи: атомарні і контейнера. До першого типу можна віднести дату, номер, назву, підпис, комірки таблиці або списку. Прикладами іншого типу є таблиці, схеми, формули і списки. Позначимо перший тип атрибутів як множина A, а другий множина B. В результаті документ можна представити як сукупність елементів, як показано у формулі (1) або (2) [8].

$$D = \{a_1, a_2, \ldots, a_i, b_1, b_2, \ldots, b_j\}, i = \overline{1, \ldots, \mathrm{N}}, j = \overline{1, \ldots, \mathrm{M}}, a_i \in \mathrm{A}, b_j \in \mathrm{B} \quad (1)$$

$$D = \{\mathrm{A}_i, \mathrm{B}_j\}, i = \overline{1, \ldots, \mathrm{N}}, j = \overline{1, \ldots, \mathrm{M}} \quad (2)$$

Атомарні властивості виявити простіше, так як зазвичай вони мають типове місце положення. Наприклад, дата та підпис в заяві знаходяться після головного тіла інформації. Завдяки цьому є можливість використання гнучких моделей, де можна отримати назву атрибуту та його значення. Якщо ж в документі наявні дані, що мають комплексний вид, то необхідно перевірити ключові елементи. Наприклад, характерні символи, такі як «=», «Σ» або інші математичні знаки, означають, що об'єктом є формула. Для схем властиво наявність різний типів фігур або малюнки. Якщо робота здійснюється з таблицею, то необхідно ідентифікувати об'єкт типу таблиці за наявністю кордонів. Якщо вони не визначені, то кожну клітинку можна визначити по межах тексту [9].

Далі інформацію ділиться за певними атрибутами або на блоки. Її зберігають у певному, наприклад XML або JSON, щоб у подальшому зручно було порівнювати та отримані результати були точними.

Останнім етапом алгоритму є порівняння. Він містить в собі 2 функції: визначення показника схожості документів та виділення тексту, що не співпадає з іншим. Першу знаходять за допомогою строковою метрикою, що представляє собою певний алгоритм, який визнає, наскільки перший текст відрізняється в другого в плані додавання, переміщення або видалення символів. Інша функція знаходить усі положення будь-яких не співпадаючих символів та виділяє їх певним кольором, в залежності від операції, яка була проведена над ними.

Схема алгоритму представлена на рисунку 1.2.

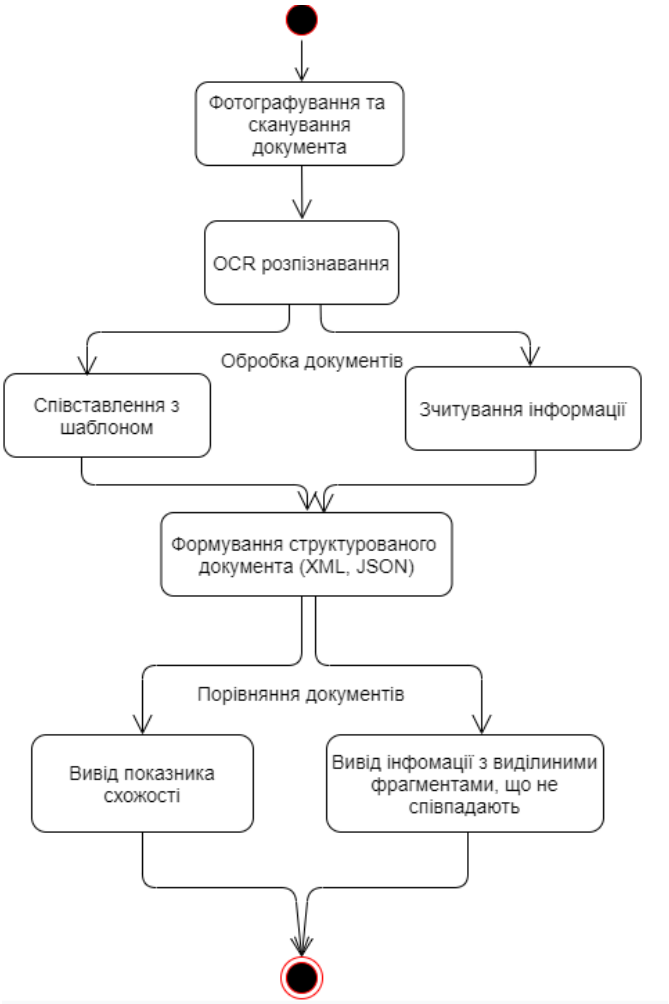


Рисунок 1.2 – Схема структурна діяльності процесу порівняння документів

1.5 Аналіз успішних ІТ-проектів

1.5.1 Аналіз відомих технічних рішень

Технологічні рішення, які б одночасно вирішували задачу розпізнавання тексту для порівнювання графічних зображень, і порівняння файлів різного формату не існує. Але є такі, що виконують кожну функцію окремі. Далі наведено декілька технологічних рішень, що мають можливість розпізнавати текст з зображень:

– Tesseract – бібліотека з відкритим кодом, для вирішення задачі використовує підхід нейронних мереж. Технологія підтримує Unicode та розпізнає більше 100 мов. Є можливість установити на будь-якій операційній

системі та використовувати з багатьма мовами програмування. Безкоштовна для використання [10].

- Iron OCR – технологія компанії Iron Software, що дозволяє зчитувати текст та штрих-код із зображень та зберігати їх у вигляді відкритого тексту або структурованих даних. Перед обробки понижу фоновий шум, працює з обертанням, спотворенням тексту, спрощує кольори та підвищу контраст, що дозволяє отримати біль точні результати. В наявності моє 22 мови, включаючи російську, але без української. Безкоштовною є відкрита версія, але для приватного або сумісного використання платна.
- CnetSDK.OCR – бібліотека, що має вирішення задачі розпізнавання тексту з більше ніж 60 мов, включаючи українську. Підтримує усі формати зображення. Використовує технологію Tesseract, але в результаті видає більш точні результати. Е безкоштовна версія, але лишена пробний період. Після його використання лише платна.
- Asprise OCR - в першу чергу це безкоштовна технологія, що розпізнає текст не тільки в зображеннях, а ще з файлів формату pdf. Є можливість розпізнавати більше 20 мов, але в безкоштовній версії надані лише англійська, французька, німецька та португальська. За інші потрібно платити, але вони не включають українську.
- ByteScout Text Recognition SDK – фреймворк, що може розпізнати текст із зображення будь-якого формату, а також зберігати формат даних. Використовує фільтри для покращення якості зображення. Розпізнає 103 мови, включаючи українську. Технологія є платною для використання, але має безкоштовну тестову версію.

В таблиці 1.2 наведені вище наведені технологічні рішення розпізнавання тексту та основні їх характеристики.

Таблиця 1.2 – Порівняння технологій розпізнавання тексту

Назва технології	Кількість мов	Наявність української мови	Повністю безкоштовна
Tesseract	103	+	+
Iron OCR	22	–	+
CnetSDK.OCR	60	–	+
Asprise OCR	20	–	+
ByteScout Text Recognition SDK	103	+	–

1.5.2 Аналіз відомих програмних продуктів

Є чимало готових продуктів, які надають можливість порівнювати документів різних форматів. Прикладами таких готових програм є:

- ABBYY Comparator – програма, що може порівнювати документи багатьох форматів, включаючи графічні зображення. Також є можливість відображення важливих невідповідностей: видалення, додавання або зміна тексту, паралельного перегляду знайдених невідповідностей в обох документах. А також здатна формувати звіти по здійсненій роботі. Фактично програма виконує необхідні функції, але вона є платною для використання та не виводить необхідну статистику.
- ABYY FineReader – програмне забезпечення, що надає можливість розпізнавати текст з фотографій та скану документа на більше ніж 190 мов, редагувати, коментувати та з’єднувати необхідну інформацію з декількох документів. А також вбудована функція порівняння файлів багатьох форматів. Програма є платною для використання.
- Docu-Proof Enterprise – також може порівнювати документи багатьох форматів, включаючи графічні зображення, здатна формувати звіти по

здійсненій роботі. Вона є платною для використання, хочу можна спробувати демо-версію. Має складний для роботи інтерфейс.

- Microsoft Word – найпопулярніший додаток користувачів операційної системи Windows. Має функцію порівняння документів, але не графічного виду. А також можуть виникнути проблеми з файлами формату pdf, так як програма конвертує їх в формат doc. Є безкоштовна та платна версії.
- Diff Doc – безкоштовний програмний продукт, що порівнює документи без можливості вибору графічних файлів. Має простий та зрозумілий інтерфейс, є можливість налаштовувати параметри роботи та редагувати файли. А також можна працювати за допомогою командної строки.
- WinMerge – безкоштовна програма з відкритим програмним кодом. Являє собою системою контролю версій, що дає можливість порівнювати не тільки файли, а також файлову систему. Є функцію порівняння графічних зображень, але без розпізнання тексту. Перевіряє на відповідність файли лише одного формату. Має простий для роботи інтерфейс.
- DiffMerge – інший приклад програми, що являє собою систему контролю версій. Безкоштовна для використання, потрібно лише зареєструватись для отримання ліцензії.

Проаналізувавши ряд вище вказаних програмних продуктів можна виявити переваги та недоліки існуючих рішень. Більшість з них мають функцію розпізнавання тексту на графічних файлах та порівнювати їх з іншими видами. В усіх є можливість відслідковувати зміни та аналізувати їх. Але, в основному, такі програми є платними, іноді є змога спробувати роботу на безкоштовній версії. Ті продукти, що не є платними, не дають потрібну функцію роботи з файлами. Тому створення безкоштовного продукту з функцією порівняння графічних та інших видів документів є актуальною.

1.6 Аналіз вимог до програмного забезпечення

1.6.1 Розроблення функціональних вимог

Діграма варінтів використання зображеня на рисунку 1.2. В таблиці 1.3 наведені функціональні фимоги та їх пріорітети.

Рисунок 1.3 – Схема варіантів використання

Таблиця 1.3 – Функціональні вимоги та їх пріоритети

Варіант використання	Функціональна вимога	Пріоритет
Завантажити файл	1. Програма має надавати можливість завантажити файл з файлової системи.	Високий

Продовження таблиці 1.3.

Розпізнати текст	2. Програма повинна розпізнавати текст на вказаних форматах зберігання документів та виводити відповідний результат про успішність або помилку. 2.1 Програма має розпізнавати текст на зображення та видавати точний результат. 2.2 Програма має зчитувати інформацію з текстових документів вказаного формату. 2.3 Програма має зберігати інформацію у зручному для порівнянні форматі (XML, JSON).	Високий
Порівняти документи	3 Програма має надавати можливість порівнювати файли графічних та інших форматів, при чому обидва файли можуть бути однакового чи різних форматів. 3.1 Програма порівнювати дані відповідно по блокам. 3.2 Програма має видавати результат порівняння у вигляді відсотку співпадіння. 3.3 Програма має надавати можливість перегляду невідповідного результату.	Високий

Продовження таблиці 1.3.

	3.4 Програма має надавати можливість зберегти результати порівняння в файлі.	

Функціональні вимоги / Варіант використання	1.Завантажити файл з файлової системи	2.Розпізнати текст	2.1 Розпізнати з зображення	2.2 Зчитати з текстового файлу	2.3 Зберегти дані в JSOM або XML	3. Порівняти файли різних і однакових форматів	3.1 Порівняти по блокам	3.2 Видати відсоток схожості	3.3 Вивести невідповідні дані	3.4 Зберегти результати
Завантажити файл	■	■								
Розпізнати текст			■	■	■					
Порівняти документи						■	■	■	■	■

Рисунок 1.4 – Схема матриці трасування

1.6.2 Розроблення нефункціональних вимог

Були виділені наступні нефункціональні вимоги:

- продукт повинен працювати на операційній системі Windows та при наявності Microsoft Word версії 2007 та вище;
- для отримання більш точного результату розпізнавання тексту зображення має бути розміром щонайменше 300 dpi;
- для розпізнавання тексту використовувати зображення форматів png, jpg, jpeg, tiff;
- для зчитування тексту з інших видів файлів, їх формат повинене бути doc, docx, dot, dotx, rtf або pdf;
- має буди передбачений захист від некоректних дій користувача;
- інтерфейс має бути простим для використання.

1.6.3 Постановка комплексу завдань модулю

Призначенням розробки є програмне забезпечення, яке розпізнає текст в сканованих або сфотографованих документах та порівнює їх з електронною копією і видає відповідний результат.

Метою розробки є підвищення ефективності роботи користувача при перевірці на відповідність документів в паперовому та електронному вигляді та зменшення ймовірності виникнення помилки при аналізі та висновка за допомогою надання зручного інтерфейсу, через який можна завантажити потрібні документи та отримати інформацію щодо їх схожості.

Для досягнення поставленої мети потрібно вирішити наступні задачі:

- визначати формат вхідних та вихідних даних;
- розробити модуль розпізнавання тексту. Модуль в першу чергу повинен надавати можливість розпізнавання українську мову та символів на зображеннях та відсканованих документах;
- створити функцію витягування інформації з файлів текстового формату. Функція має зчитувати дані таких форматів, як doc, docx, rtf, dot, dotx, pdf, txt;
- створити модуль виводу інформації у зручному для читання та редагування форматі. Модуль має надавати можливість зберігання інформації в зручному для читання та редагування за необхідності форматі;
- створити функцію підрахування проценту відповідності двох документів. Функція має визначати наскільки файли схожі та видавати точний результат для аналізу;
- розробити модуль демонстрації невідповідних даних. Модуль повинен вивести дані тексти та виділити інформацію, що не співпадає;
- створення зручного інтерфейсу користувача.

Висновки по розділу

В результаті проведення аналізу вимог до розробки програмного забезпечення проведено дослідження предметного середовища – OCR технологія. На даний момент вона є розповсюдженою у використанні. Технологія не завжди дає точний результат, але виконуються розробки та вдосконалюються алгоритми для вирішення цієї проблеми.

Досліджено алгоритм розрахування відстані Дамерау-Левенштейна, який є модифікованим варіантом розрахування відстані Левенштейна. Визначені його переваги та недоліки.

Крім того досліджено опис процесів роботи алгоритму порівняння документів, який складається з таких етапів, як завантаження документів, розпізнавання або зчитування тексту. Потім маємо обрубку документів, що складається з кроку співставлення з шаблоном та витяг інформації, далі формування даних в певний формат та визначення відмінностей. Останній етап ділиться ще на два кроки: це знаходження показника схожості за певним алгоритмом та виділення різних частин текстів. Також створено сценарій користувача, визначені функціональні та нефункціональні вимоги.

Проведено дослідження готових технологічних рішень (Tesseract, Iron OCR, CnetSDK.OCR, Asprise OCR, ByteScout Text Recognition SDK) і програмних продуктів (ABBYY Comparator, ABBYY FineReader, Docu-Proof Enterprise, Microsoft Word, Diff Doc, WinMerge, DiffMerge). Виявлено, що більшість з них мають функцію розпізнавання тексту на графічних файлах та порівнювати їх з іншими видами, але, в основному, такі програми є платними. Інші ж продукти не дають потрібну функцію роботи з файлами. Тому було вирішено створити безкоштовний продукт з функцією порівняння графічних та інших видів документів.

Сформульовані головні задачі розробки програмного забезпечення: розпізнати текст, зберегти його в зручному для порівняння форматі, виконати порівняння та вивести результати.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Загальні процеси, які проходить користувач, включають процес завантаження файлів, обробка графічних документів та отримання інформації порівняння.

Послідовний опис етапу завантаження файлів:

- користувач вводить шлях до файлу або шукає в своїй файловій системі. У разі успішного знаходження переходить до вибору другого файлу.
- користувач вибирає спосіб завантаження другого файлу та вводить шлях до нього. У разі успішного пошуку переходить до вибору інших властивостей документів.
- користувач вибирає мову, на якій написана інформація в документах, та їх тип. У разі успішного пошуку переходить до обробки файлів.

На етапі обробки документа користувач лише натискає на кнопку «Отримати текст» для необхідного файлу. У разі успішної операції отримуються тексти документів у форматі JSON у відповідних текстових полях. Користувач перевіряє правильність виведеної інформації.

Послідовний опис етапу порівняння:

- після обробки файлів користувач натискає кнопку «Порівняти». У разу успішного виконання операції він отримує відповідь у вигляді статистики відповідності файлів відносно певних блоків файлу JSON.
- користувач натискає на кнопку «Показати різницю текстів» за бажанням переглянути степінь відповідності більш детально. У разі успішного виконання операції виділяються усі невідповідності документів.

Схема бізнес-процесу роботи даного програмного забезпечення представлена у графічному матеріалі дипломного проекту.

2.2 Архітектура програмного забезпечення

Програмне забезпечення складається з 3 основних модулів, кожні з яких відповідають своїм задачам:

- модуль розпізнавання тексту;
- модуль порівняння документів;
- модуль інтерфейсу.

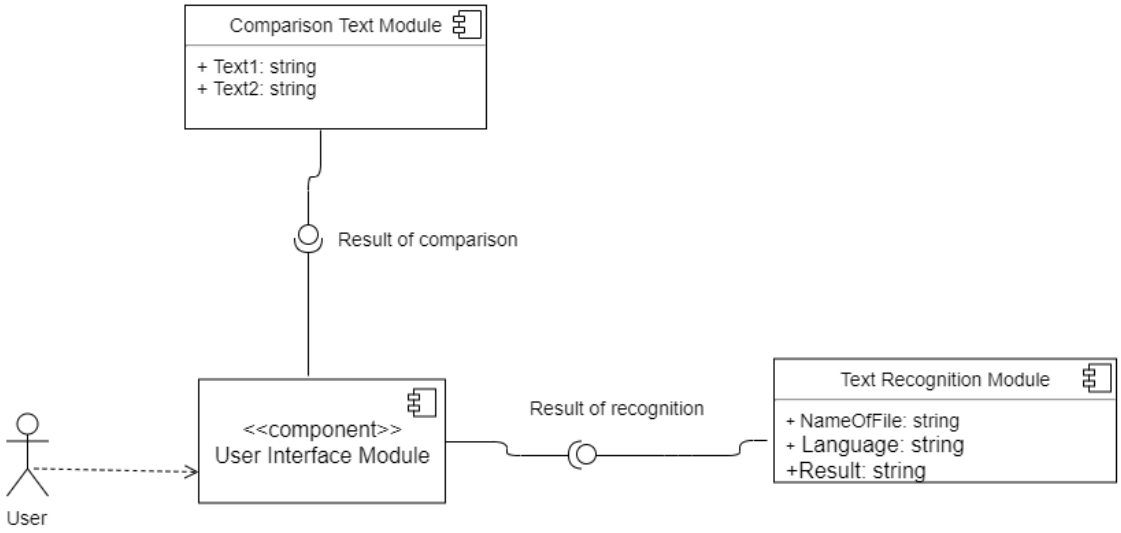


Рисунок 2.2 – Схема архітектурних компонентів

Для розробки програми було вирішено використовувати мову програмування C# для написання функціональних модулів, для проектування інтерфейсу користувача використовується Windows Form, а для розпізнавання технологію Tesseract.

2.2.1 Опис архітектури програмного забезпечення

Програма містить в собі такі модулі: Text Recognition Module, User Interface Module, Comparison Module. Кожен з них відповідає за певну частину функціоналу. В таблиці 2.1 описані кожен модуль та класи, що до нього входять.

Таблиця 2.1 – Опис спроектованих модулів та відповідних класів

Назва модуля	Клас	Опис та призначення класу
Text Recognition Module	FileManager	Клас, що відповідає за виклик відповідного класу для зчитування інформації. Отримує назву файлу, визначає тип файлу та передає дані наступному класу.
	ReadImgFile	Виконує функцію розпізнавання тексту в графічних зображеннях з такими форматами, як png, jpg, jpeg. Отримує назву файлу та мову інформації і видає розпізнаний текст.
	ReadTextClass	Виконує функцію зчитування тексту з файлів формату txt. Отримує назву файлу і видає зчитаний текст.
	ReadWordFile	Виконує функцію зчитування тексту з файлів формату doc, docx, dot, dotx, rtf. Отримує назву файлу і видає зчитаний текст.

Продовження таблиці 2.1.

	PdfToImg	Виконує обробку та форматування файлу формату pdf в графічний формат png, jpg, jpeg, tiff для зручного розпізнання тексту. Отримує назву файлу та повертає зображення або набір зображень.
	PdfReadFile	Виконує функцію зчитування тексту з файлів формату pdf. Отримує зображення або набір зображень, які є результатами роботи класу PdfToImg і видає зчитаний текст.
	SplitOnBlocks	Містить шаблони різних типів документів. Отриману інформацію співставляє з шаблонами та розподіляє на блоки відповідно до них. Повертає інформацію для конвертування його у JSON формат.
	ConvertJS	Отримує дані вже в розподіленому форматі та виконує функцію серилізації та десерилізацію для формату JSON.

Продовження таблиці 2.1.

User Interface Module	Form1	Відображає форму та все елементи керування, які доступні для використання. Відповідає за роботу кожного елементу та реагує на подіє, які виникають з кожним з них. Містить в собі програмну логіку.
	Program	Запускає програму, що описана в класі Form1.
Comparison Module	ComparisonTwoText	Реалізовує алгоритм знаходження відстані Дамерау-Левенштейна, результат якого являє собою процент відповідності текстів. Отримує два тексти для порівняння, видаляє зайві чисті строки та відступи і підраховує відстань. В результаті повертає дійсне число.
	OutputDifference	Виводить виділяючи усі не співпадіння текстів. Отримує два тексти для порівняння, знаходить усі відмінності та виводить їх зафарбованими відповідними кольорами.

Продовження таблиці 2.1.

	SaveResult	Зберігає дані показників схожості та інформацію документів з виділеним текстом, що відрізняється.

2.2.2 Архітектурні компоненти програмного забезпечення

Модуль розпізнавання тексту.

Перед серилізацією та десерилізацією інформації в формат JSON [11] використовується проміжна сутність, яка описана в класі FileBlock. Об’єкт саме цього класу передається в функції для перетворення в потрібний формат, щоб в подальшому порівнювати результати. Дані, що розпізнаються з документа, розподіляються на 3 блоки. Опис сутності наведений в таблиці 2.2.

Таблиця 2.2 - Опис класу FileBlock

Назва властивості	Тип	Опис
Header	string	Містить в собі верхню частину документи. Сюди можуть входити ключові слова, що класифікують файл. Наприклад, слово «Заява» відносить даний документ до класу заяв.
Body	string	Містить головну інформацію, яка розміщується в документі. Не містить ключових слів, так як в кожному файлі вона відрізняється. Знаходиться шляхом визначення інших блоків та методом виключення.

Продовження таблиці 2.2.

Footer	string	Розміщує нижню частину документа. Так може характеризувати файл за ключовими словами. Наприклад, слово «СУМА» може класифікувати файл як чек.

Даний модуль займає значну частину проекту, так як реалізовано функції не тільки розпізнавання тексту з зображення, а й зчитування інформації з документів інших можливих форматів. Головним класом, який напряму взаємодіє з формою відображення, є FileManager. Він отримує введені користувачем назви файлів, в залежності від визначеного формату класифікує документ та передає для подальшої роботи з ним відповідному класу. Далі виконується зчитування інформації. Цей процес залежить від класу файлу, наприклад, усі документі, що створені Microsoft Word 2007 або пізнішою версією, відкриваються та розпізнають текст за допомогою бібліотеки Microsoft.Office.Interop.Word [12]. Для правильної роботи класу наявність програми Microsoft Word 2007 версії або пізніше необхідно. Для правильної роботи функціоналу та отримання точних результатів необхідно вибрати мову інформація, яка є в документі та завантажити файл в хорошому форматі. Чим кращий формат зображення, тим точніше буде отриманий текст і ймовірність помилок зменшиться. Зчитування файлів формату pdf виконується в 2 етапи: обробка та перетворення даного файлу в зображення, а потім виконується функція розпізнавання тексту, так як готових технологічних рішень з безкоштовним використанням та наявністю української мови немає. Для виконання функції перетворення формату pdf в зображення використовується бібліотека GhostscriptSharp. Опис основних методів класів наведений в таблиці 2.3.

Таблиця 2.3 - Опис класів та їх методів модулю розпізнавання тексту

Назва класу	Метод	Параметри Тип значення, що повертається	Опис
FileManager	GetExtension()	string NameOfFile string extension	Отримує назву файлу, визначає його формат та повертає його у вигляді строки.
	GetFileType()	string extension FileType type	Отримує строку формату файлу та визначає його тип.
	ManageFile()	FileType type void	В залежності від типу файлу викликає відповідний клас для зчитування інформації і повертає текст результату.
ReadImgFile	Read()	string NameOfFile, string Language string Result	Виконує функцію розпізнавання тексту із зображення. Для цього використовується технологія Tesseract.
ReadTextClass	Read()	string NameOfFile string Result	Виконує зчитування тексту з формату txt. Використовує звичайний файловий потік File IO.

Продовження таблиці 2.3.

ReadWordFile	Read()	string NameOfFile string Result	Виконує функцію зчитування інформації з файлів, створених програмою Microsoft Word версії 2007 або пізніше. Використовується бібліотека Microsoft.Office.Interop.Word.
SplitOnBlocks	SplitOnLine()	string textResult void	Видаляє усі зайві рядки та пропуски та створює список рядків відповідного тексту.
	CheckTemplate()	List<string> lines FileBlock	Отримує список усіх рядків тексту, використовує шаблон типу документу «чек». Знаходить ключові слова верхнього та нижнього блоку та ділить і додає відповідні рядки до властивостей класу FileBlock.

Продовження таблиці 2.3.

	ArticleTemplate( )	List<string> lines FileBlock	Отримує список усіх рядків тексту, використовує шаблон типу документу «стаття». Знаходить ключові слова верхнього та нижнього блоку і додає відповідні рядки до властивостей класу FileBlock.
	ApplicationTem plate()	List<string> lines FileBlock	Отримує список усіх рядків тексту, використовує шаблон типу документу «заява». Знаходить ключові слова верхнього та нижнього блоку і додає відповідні рядки до класу FileBlock.
ConvertJS	ConvertToJS()	FileBlock file string	Виконує серилізацію об’єкта типу FileBlock та повертає текст в форматі JSON.

Продовження таблиці 2.3.

	ConvertFromJS( )	string textResult string	Виконує десерилізацію тексту формату JSON та повертає інформацію в читабельному вигляді.
PdfToImg	GetCountOfPages()	string NameOfFile int	Відкриває файл із вказаною назвою, підраховую кількість сторінок та повертаю це значення для подальшої роботи.
	ConvertPdfToImg()	string NameOfFile, int countOfPages void	Використовує модуль GhostscriptSharp для перетворення документу в формат зображення із вказаною назвою. Використовується вихідний файл лише тимчасово, щоб отримати текст.
ReadPfdFile	Read()	string NameOfFile string Result	Виконує зчитування тексту з формату pdf. Використовує технологію Tesseract.

Розпізнавання тексту з графічних зображень виконується за допомогою технології Tesseract. Вона була вибрана для використання та рішення проблеми через можливість розпізнавання багатьох мов, особливо української, та безкоштовне застосування. Бібліотека Microsoft.Office.Interop.Word використовується для зчитування інформації з текстових файлів таких форматів, як doc, docx, dot, dotx, rtf. Бібліотека GhostscriptSharp використовується для роботи з файлами формату pdf [13]. Опис функцій, що використовуються в даному модулі із вище наведених бібліотек, вказаний в таблиці 2.4.

Таблиця 2.4 – Опис головних функцій з використаних бібліотек.

Бібліотека	Функція	Параметри Вихідні дані	Опис
Tesseract	TesseractEngine()	string pathToTessdata, string Language string Result	Використовує файли з даними, які знаходяться в pathToTessdata, щоб за вказаною мовою розпізнати текст в зображені та поверну у вигляді рядку для подальшої роботи.
GhostscriptSharp	GeneratePageThumbs()	string NameOfFile, int countOfPages image file	Створює зображення із вказаною назвою та копіює усі сторінки з файлу формату pdf.

Продовження таблиці 2.4.

Microsoft.Office.Interop.Word	Documents.Open()	string NameOfFile	Відкриває файл, що створений Microsoft Word версії 2007 або більше, для подальшого зчитування інформації. Після виконаної роботи закриває файл.

Модуль порівняння документів.

Даний модуль виконує функції порівняння двох текстів та повертає відповідний результат. В класі CompareTwoText виконується обробка тексту, видаляються зайві чисті строки та відступи. Відсоток співпадінь визначається за допомогою алгоритму розрахування відстані Дамерау-Левенштейна. Клас SaveResult виконує роботу збереження інформації в файл з відповідною назвою та формату rtf. Дані зберігаються з певним шрифтом та кольором в такому порядку: спочатку вказуються показники схожості відповідних блоків, потім розпізнана інформація першого документа з виділеним текстом, якщо є, потім текст другого документа з виділеннями, якщо такі існують. Опис методів даного модулю представлений в таблиці 2.5.

Таблиця 2.5 - Опис класів та методів модулю порівняння

Назва класу	Метод	Параметри Тип значення, що повертається	Опис
ComparisonTwoText	SplitString()	string text1, string text2 void	Метод видаляє зайві чисті строки та відступи.

Продовження таблиці 2.5.

	Min()	int a, int b, int c, int d int	Знаходить та повертає мінімальне значення з вказаних параметрів.
	ComparingText()	string text1, string text2 double	Метод являється реалізацією алгоритму знаходження відстані Дамерау-Левенштейна
OutputDifference	OutPutDiffText()	string text1, string text2 void	Виводить текст, при цьому виділяючи усі не співпадаючі елементи.
SaveResult	GetNameOfFile()	string nameOfFile1, string nameOfFIle2 string	Створює назву документа. З’єднуючи назви обох документів та записує між ними слово «порівняти».

Продовження таблиці 2.5.

	WriteResult	RichTextBox box1, RichTextBox box2, RichTextBox box 3 void	Копіює всю інформацію разом зі шрифтом та кольором в окремий об’єкт типу RichTextBox та записує його дані потрібного формату в текстовий файл. Якщо дані записані успішно, то виводиться повідомлення.

Модуль інтерфейсу користувача. Інтерфейс користувача розроблений за допомогою Windows Form. Він складається лише с 2 класів: Form1, Program. На формі знаходяться такі елементи керування, як button, richtextbox, combobox, label та textbox. Опис методів даного модулю представлений в таблиці 2.6.

Таблиця. 2.6 - Опис класів та методів модулю інтерфейсу

Назва класу	Метод	Параметри Тип значення, що повертається	Опис
Form1	button1_Click()	Object sender, EventArgs e void	Реагує на натиск кнопки «Отримати текст» першого файлу та виконує розпізнавання тексту і виводить результат у відповідне поле.
	button2_Click()	Object sender, EventArgs e void	Реагує на натиск кнопки «Отримати текст» другого файлу та виконує розпізнавання тексту і виводить результат у відповідне поле.
	button5_Click()	Object sender, EventArgs e void	Реагує на натиск кнопки «Порівняти» та виконує порівняння тексту і виводить результат у відповідне поле.

Продовження таблиці 2.6.

	button6_Click()	Object sender, EventArgs e void	Реагує на натиск кнопки «Показати різницю текстів» другого файлу та тексти з підкресленими невідповідностями у відповідне поле.
	button4_Click()	Object sender, EventArgs e void	Реагує на натиск кнопки «Завантажити файл з системи» другого файлу та заносить назву файлу у відповідне поле.
	button3_Click_1()	Object sender, EventArgs e void	Реагує на натиск кнопки «Завантажити файл з системи» першого файлу та заносить назву файлу у відповідне поле.

Продовження таблиці 2.6.

	button7_Click()	Object sender, EventArgs e void	Реагує на натиск кнопки «Зберегти результати». Зберігає результати порівняння в текстовий файл.
Program	Main()	void	Запускає клас Form1.

Основні використані елементи керування інтерфейсом є [14]:

- Form – головний елемент інтерфейсу програмного забезпечення, на якому розташовуються всі інші елементи;
- button – елемент інтерфейсу кнопка, який при натисканні починає виконувати функції, що написані в програмі;
- label – елемент інтерфейсу, який лише виводить інформацію, немає можливість редагувати на формі;
- textbox – елемент інтерфейсу, що записує, редагує та виводить інформацію в залежності з роботи програми;
- richtextbox - елемент інтерфейсу, що записує, редагує та виводить інформацію в залежності з роботи програми, відрізняється в textbox заданим розміром елементу;
- combobox – елемент інтерфейсу, який дає можливість перегляду списку та вибирати потрібний елемент.

2.3 Аналіз безпеки даних

Для безпеки даних від перегляду сторонніх користувачів можна покращити програмне забезпечення шляхом доповнення функції завантаження

файлу з системи. Це можна зробити за допомогою шифрування AES-GSM доступу до файлів, щоб лише власник ключа мав змогу відрити відповідний документ для зчитування інформації. Іншим варіантом є використання мастер-ключа для блокування всього файлового середовища, та лише його власник може відкривати доступ.

Висновки по розділу

На даному етапі розробки продукту змодельовано сценарій використання програми користувач та створено BPMN діаграму.

Також визначено 3 головні модулі програмного забезпечення: модуль розпізнавання тексту, модуль порівняння документів та модуль інтерфейсу користувача.

Модуль розпізнавання тексту надає функція зчитування інформації як із зображення, так і файлів формату doc, docx, dot, dotx, rtf, pfd. В результаті отримана інформація зберігається в форматі JSON для подальшої роботи з нею. Модуль використовую технологію Tesseract та Microsoft.Office.Interop.Word для зчитування даних з файлів, створених програмою Microsoft Word.

Модуль порівняння тексту отримує дані відповідного формату та порівнює кожну частину окремо від інших. Реалізована функція розрахунку відстані Дамерау-Левенштейна, котра виводить результати знаходження показника схожості документів. Також реалізований клас, що відповідає за виведення тексту та виділення фрагментів, що відрізняються.

Модуль інтерфейсу надає користувачу елементи керування програмою та виконує функціонал усі інших модулів в залежності від дій користувача.

Під час аналізу безпеки даних досліджено та запропоновано методи захисту інформації від сторонніх користувачів.

Програмне забезпечення надає можливість розпізнавання тексту з таких форматів зображення, як png, jpg, jpeg, tiif, та зчитує інформацію з текстових файлів форматів doc, docx, dot, dotx, rtf, pdf. Також виконує функцію

порівняння документів як однакових, так і різних форматів и видає відповідний результат. Продукт надає користувачу зручний та зрозумілим для використання інтерфейс.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування програмного забезпечення є необхідною частиною розробки продукту. За допомогою послідовного виконання тестів можна перевірити функціональну поведінку програми, зручність використання та правильність роботи функціоналу.

Перевагами тестування ПЗ є:

- знаходження та виправлення помилок до того, як продуктом почне користуватись аудиторія;
- мінімізація технічних втрат та швидкий розвиток процесу розробки через виконання тестування на декількох стадіях створення;
- оптимізація коду.

Розробка комп'ютерних програм має свої привілеї, так як не потрібно піклуватись про розмір екранів та через більшу потужність та швидкість обробка даних проходить швидше порівнюючи з іншими пристроями. Проте з іншої сторони це не означає, що процес тестування дуже простий.

В першу чергу потрібно проаналізувати та виконати тести роботи програмного забезпечення на різних операційних системах. Потім перевірити роботу з різними версіями інших програмних продуктів.

При дослідженні процесу тестування програми порівняння також необхідно ретельно перевірити роботу функціоналу, тому що продукт працює напряму з документами та будь-які некоректні зміни можуть привести до певних наслідків.

Враховуючи аналіз проведення тестування комп'ютерних програм та усі складності його виконання, робота розробки продукту може йти по графіку або навіть прискоритись, кінцевий результат надасть змогу користувачам без проблем використовувати продукт.

3.2 Опис процесів тестування

Тестування виконується зазвичай на декількох етапах життєвого циклу, коли розробляється програмне забезпечення. Технологія тестування програмного забезпечення містить такі етапи:

- визначення функціоналу, що підлягає та не підлягає тестуванню;
- формулювання підходів, що будуть використовуватись для даного продукту;
- написання тест кейсів;
- розробка критерію проходження тестів;
- визначення вимог середовища проведення тестування;
- проведення тестування та оцінка результатів;
- звітність результатів.

Системні вимоги:

- операційна система Windows 7 або наступний версій;
- наявність програми Microsoft Word версії 2007 або наступних;
- наявність програми Adobe Acrobat Reader DC версії 5.1 або наступних.

Апаратні вимоги:

- достатньо вільної пам’яті комп’ютері для встановлення програми;
- 32 або 64-розрядний процесор з тактовою частотою 1GHz;
- оперативна пам’ять не менше 1Gb.

Тестування проводилось на ноутбуці компанії DELL з процесором Intel® Core i3-6060U, тактовою частотою 2GHz, операційною системою Windows 10 та при наявності програми Microsoft Word 2016 та програми Adobe Acrobat Reader DC версії 5.1.

3.3 Опис контрольного прикладу

Під час тестування даного програмного забезпечення було проведено перевірку усіх функціональних вимог та варіантів використання, враховано

роботу інтерфейсу та перевірено усі запобіжні функції від некоректних дій користувача. Результати тестів наведені в таких таблицях:

- завантаження документа з файлової системи (таблиця 3.1);
- перевірка роботи при не вказаних властивостях документа (таблиця 3.2);
- розпізнавання тексту з файлу зображення (таблиця 3.3);
- зчитування інформації з текстового формату (таблиця 3.4);
- розрахунок показника схожості документів (таблиця 3.5);
- перевірка роботи порівняння відсутності одного з тексту (таблиця 3.6)
- виділення не співпадаючої інформації (таблиця 3.7);
- збереження інформації в текстовому файлі (таблиця 3.8).

Таблиця 3.1 - Завантаження файлу з файлової системи

Мета тесту	Перевірити роботу функції завантаження назви файлу з системи.
Початковий стан	Програма запущена. Видно інтерфейс користувача.
Вхідна дані	-
Проведення тесту	Натиснути кнопку із зображенням папки для першого файлу, вибрати потрібний файл з системи, натиснути кнопку «Відкрити».
Очікуваний результат	В текстовому полі під назвою «Назва першого документу» виведеться рядок повно шляху та назви файлу.
Фактичний результат	В текстовому полі під назвою «Назва першого документу» вивівся рядок повно шляху та назви файлу.

Таблиця 3.2 – Перевірка роботи при не вказаних властивостях документа.

Мета тесту	Перевірити роботу виведення повідомлення про некоректну обробку тексту через відсутність параметрів.
Початковий стан	Обраний файл з файлової системи.
Вхідна дані	Назва першого документа.
Проведення тесту	Натиснути кнопку «Отримати текст» для першого документа.
Очікуваний результат	Виведеться окреме вікно з повідомленням «Виберіть мову документів!». Потім виведеться інше повідомлення «Виберіть тип документів!»
Фактичний результат	Вивелося окреме вікно з повідомленням «Виберіть мову документів!». Потім вивелося інше повідомлення «Виберіть тип документів!»

Таблиця 3.3 - Розпізнавання тексту з файлу зображення

Мета тесту	Перевірити роботу функції розпізнавання тексту із зображення та точні її результату.
Початковий стан	Вибраний перший документ для розпізнавання. Вибрана мова даних. Вибраний тип документа.
Вхідна дані	Назва файлу, мова, тип.
Проведення тесту	Натиснути кнопку «Отримати текст» для першого документа.
Очікуваний результат	В лівому текстовому полі виводиться розпізнаний текст у форматі JSON. Усі слова розпізнанні без помилок.
Фактичний результат	В лівому текстовому полі виводиться розпізнаний текст у форматі JSON. Усі слова розпізнанні без помилок.

Таблиця 3.4 – Зчитування інформації з текстового файлу

Мета тесту	Перевірити роботу функції зчитування даних з файлу текстового формату та точні її результату.
Початковий стан	Вибраний перший документ для розпізнавання. Вибрана мова даних. Вибраний тип документа.
Вхідна дані	Назва файлу, мова, тип.
Проведення тесту	Натиснути кнопку «Отримати текст» для першого документа.
Очікуваний результат	В лівому текстовому полі виводиться розпізнаний текст у форматі JSON. Усі слова розпізнанні без помилок.
Фактичний результат	В лівому текстовому полі виводиться розпізнаний текст у форматі JSON. Усі слова розпізнанні без помилок.

Таблиця 3.5 - Розрахунок показника схожості файлів

Мета тесту	Перевірити роботу функції розрахунку показника схожості.
Початковий стан	Обрані 2 документи для порівняння, зчитані та показані обидва тексти.
Вхідна дані	Текст першого файлу та другого у форматі JSON.
Проведення тесту	Натиснути кнопку «Порівняти».
Очікуваний результат	В текстовому полі під назвою «Відсоток схожості» виведеться інформація схожості по блокам даних. Для Header: 100%, Body: 100%, Footer: 92,4386%.
Фактичний результат	В текстовому полі під назвою «Відсоток схожості» вивелася інформація схожості по блокам даних. Для Header: 100%, Body: 100%, Footer: 92,4386%.

Таблиця 3.6 – Перевірка роботи порівняння відсутності одного з тексту.

Мета тесту	Перевірити роботу виведення помилки при некоректному порівняні документів.
Початковий стан	Обраний та розпізнаний текст лише одного документа. Обрані мова та тип.
Вхідна дані	Текст першого документа у форматі JSON.
Проведення тесту	Натиснути кнопку «Порівняти».
Очікуваний результат	У додатковому вікні виведеться попередження «Необхідно 2 документи для порівняння!»
Фактичний результат	У додатковому вікні вивелося попередження «Необхідно 2 документи для порівняння!»

Таблиця 3.7 - Виділення не співпадаючої інформації

Мета тесту	Перевірити роботу функції виведення та виділення неспівпадаючої інформації.
Початковий стан	Обрані 2 файли, виконані функції зчитування інформації.
Вхідна дані	Тексти першого та другого документів у форматі JSON.
Проведення тесту	Натиснути кнопку «Показати різницю текстів».
Очікуваний результат	В лівому текстовому полі виведеться текст без виділення, в правому виведеться текст з виділеним блоком Footer.
Фактичний результат	В лівому текстовому полі вивівся текст без виділення, в правому вивівся текст з виділеним блоком Footer.

Таблиця 3.8 - Збереження результатів в текстовому файлі

Мета тесту	Перевірити функцію збереження виведених результатів в текстовому форматі.
Початковий стан	Виведені результати показника схожості та виділені символи, що відрізняються в документах.
Вхідна дані	Результати показника схожості та обидва тексти з виділеними символами.
Проведення тесту	Натиснути кнопку «Зберегти результати».
Очікуваний результат	Створиться текстовий файл та запишуться усі результати порівняння.
Фактичний результат	Створився текстовий файл та усі результати порівняння записалися.

Висновки по розділу

Досліджено аналіз якості програмного забезпечення. Виявлено, що для даного продукту потрібно протестувати роботу інтерфейсу користувача та функціональні вимоги, так як обробка документів повинна видавати найбільш точні результати.

Окрім цього вказано опис процесу тестування, сформульовані системні та апаратні вимоги, що є необхідними для проведення даного етапу розробки продукту. Наведені вимоги до технічного та програмного забезпечення, на якому будуть проводитися тести.

Також детально описані проведені тести-кейси у вигляді таблиць. Вказано інформацію про мету тестування, початковий стан програми, вхідні дані, вказані кроки проведення тестів, а також очікуваний та фактичний результат. Проводилось тестування усіх функціональних вимог програмного забезпечення та робота виведення повідомлень на некоректні дія користувача.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання програмного забезпечення необхідно мати комп’ютер чи ноутбук із встановленою операційною системою Windows 7 або наступних версій. Також мають бути встановлені такі програми, як Microsoft Word версії 2007 або наступних для роботи з форматами таких документів, як doc, docx, rtf, dot, dotx, та Adobe Acrobat Reader DC 5.1 або наступних версій для зчитування інформації з файлів формату pdf. Для запуску програми необхідно запустити файл Comparator.exe. Перед роботою необхідно ознайомитись з інструкцією користувача, щоб дізнатись усі функціональні можливості продукту та звернути уваги на правила роботи з документами. Після цього можна одразу переходити до роботи порівняння.

4.2 Робота з програмним забезпеченням

Щоб працювати з готовим продуктом, необхідно на початку визначити, які документи будуть використовуватись для порівняння. Потрібно зауважити, що розпізнавання тексту можливе українською, англійською та російською мовами. Після цього йде завантаження файлів в систему та зчитування інформації. Після того, як вивели потрібні тексти та перевірки на помилки, виконується порівняння по блоках інформації та виведення необхідних даних. Детальний опис разом із зображеннями екрану знаходиться в додатку «Керівництво користувача».

Висновки по розділу

В даному розділі детально описано розгортання програмного забезпечення на комп’ютері користувача та наведені системі вимоги, необхідні для правильного виконання функціональних вимог програми.

Окрім цього наведено загальні кроки користування продуктом. Для ознайомлення більш детального опису етапів використання програмного забезпечення необхідно переглянути «Керівництво користувача».

ВИСНОВКИ

Дипломний проект присвячений розробці програмного забезпечення порівняння електронних та сканованих документів.

Тема є актуальною, так як більшість людей переходять на електронний документообіг, але при перевірці ідентичності інформації на паперовому та електронному версіях документа виникає проблема в правильності визначення результатів робітником.

В пункті «Аналізу вимог до ПЗ» наведено загальні положення та опис предметного середовища. Визначено що є оптичним розпізнаванням тексту та порівняно такі підходи до розпізнання, як шаблонний метод, статистичний, структурний метод, метод нейронних мереж.

Досліджено математичне забезпечення до даної розробки, роботу алгоритму розрахунку відстані Дамерау-Левенштейна. Даний метод обраний для створення функціоналу порівняння, тому що видає більш точні результати, порівняно з іншими алгоритмами через наявність допоміжного параметру. Досліджено і описано діяльність процесу порівняння документа та створена схема, що відображає її.

Також в даному пункті описаний огляд існуючих технічних рішень та готових продуктів. В результаті виявилось, що безкоштовних програмних засобів, які мають можливість розпізнавати український текст не існує. Такий висновок робить створення програми на дану тему актуальною.

Окрім цього сформовані усі функціональні та нефункціональні вимоги до продукту, варіанти використання, головна мета, призначення та цілі розробки програмного забезпечення.

В пункті «Моделювання та конструювання ПЗ» поетапно описано модель роботи користувача з програмою та розроблена схема бізнес-процесу роботи даного продукту.

Окрім цього розроблена архітектура програмного забезпечення. В результаті спроектовано 3 модулі: модуль розпізнавання тексту, модуль

порівняння та інтерфейс користувача. Наведено схему програмних компонентів.

В підрозділі «Опис архітектури ПЗ» вказано опис кожного модуля та наведені класи, з яких вони складаються. В підрозділі «Архітектурні компоненти ПЗ» міститься детальна інформація по функціональним модулям, з яких створена програма. «Модуль розпізнавання тексту» містить інформацію про сутність, які містить опис формату JSON. В такий формат записується розпізнаний текст та в подальшому порівнюється. Також наведений детальний опис класів, що відносяться до даного модулю, їх головні функції разом з вказаними параметрами та типом значення, що повертається. Вказані технологічні рішення та бібліотеки, їх функції, що використовуються для відтворення функціональних вимог. «Модуль порівняння документів» описує детально класи та їх функції, що виконують розрахунок показника відстані Дамерау-Левенштейна та виділяє усі символі, що не співпадають в обох документах. «Модуль інтерфейсу користувача» містить інформацію про всі об’єкти форми, з якими працює користувач та описує детально роботу кнопок.

Також проведено аналіз безпеки даних та виявлено, що покращенням роботи даного програного забезпечення є додавання шифрування файлів та доступу до файлової системи під час завантаження.

В розділі «Аналіз якості та тестування ПЗ» досліджено аналіз якості програми та виявлено головні моменти, які є характерними при тестування комп’ютерної програми, що працює з документами.

Також описано процес тестування, вказані системні та апаратні вимоги, що є необхідними для проведення даного етапу розробки продукту. Наведені вимоги до технічного та програмного забезпечення, на якому будуть проводитися тести.

Окрім цього описані детально проведені тести-кейси у вигляді таблиць. Вказано інформацію про мету тестування, початковий стан програми, вхідні

дані, вказані кроки проведення тестів, а також очікуваний та фактичний результат.

В розділі «Впровадження та супровід ПЗ» наведено детальний опис розгортання програми на комп’ютері користувача та вказані загальні етапи роботи з нею. Для ознайомлення детального опису користування програмним забезпечення необхідно переглянути «Керівництво користувача».

Перспективою подальшої розробки є розробка інтерфейсу для інтеграції з іншими програмами та впровадження шифрування документів, завантаження файлів з Інтернету та додавання нових типів документів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Optical character recognition [Електронний ресурс]// https://en.wikipedia.org – 2019 - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/ Optical_character_recognition
2. Lecture object recognition [Електронний ресурс]// http://cmp.felk.cvut.cz – 2004 - Режим доступу до ресурсу: http://cmp.felk.cvut.cz/cmp/courses/dzo/resources/lecture_object-recognition.pdf
3. Goswami Richa, Sharma O.P. A review character recognition techniques – International Journal of Computer Applications (0975-8887) - №7, 2013 – c.21-22
4. Dr.B.Rama, Santosh Kumar Henge OCR-The2 layered approach for classification and identification of telugu hand written mixed consonants and conjunct consonants by using advanced fuzzy logic controller // Department of Computer Science, Kakatiya University, Warangal,India – 2016 – c.77-78
5. Лавренюк М.С., Новіков О.М. Огляд методів машинного навчання для класифікації великих обсягів супутникових даних - System Research & Information Technologies, 2018, № 1. - С. 54-57
6. Огляд підходів розпізнавання машинописних текстів [Електронний ресурс]// http://asu.kpi.ua – 2019 - Режим доступу до ресурсу: http://asu.kpi.ua/wp-content/uploads/2019/05/ISTU-2019-16-17.05.pdf
7. Расстояние Дамерау-Левенштейна [Електронний ресурс]// https://ru.wikipedia.org – 2019 - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Расстояние_Дамерау_—_Левенштейна
8. Губин В.А., Гвоздинский А.Н. Объектное представление электронных текстовых документов // Радиоэлектроника и информатика. –2007. – № 1 (36). – с. 61-63
9. Олейник А.А. Алгоритм обработки слабоструктурированных бумажных документов / Матеріали XIV міжнародної наукової конференції «Інтелектуальні системи прийняття рішень та прикладні аспекти інформаційних технологій» (ISDMIT’2019), (21-25 травня 2019 року, Залізний

Порт). // Херсон: Видавництво Херсонського національного технічного університету, 2019.

10. Tesseract [Електронний ресурс]// https://github.com – 2019 - Режим доступу до ресурсу: https://github.com/tesseract-ocr/tesseract

11. JSON: What It Is, How It Works, How To Use It [Електронний ресурс]// https://www.copterlabs.com – 2019 - Режим доступу до ресурсу: https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/

12. Microsoft.Office.Interop.Word Namespace [Електронний ресурс]// https://docs.microsoft.com – 2019 - Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/api/microsoft.office.interop.word?view=word-pia

13. GhostscriptSharp [Електронний ресурс]// https://github.com – 2019 - Режим доступу до ресурсу: https://github.com/mephraim/ghostscriptsharp

14. Windows Form [Електронний ресурс]// https://docs.microsoft.com – 2019 - Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/winforms/index

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОРІВНЯННЯ ЕЛЕКТРОННИХ ТА СКАНОВАНИХ ДОКУМЕНТІВ

Технічне завдання

КПІ.ІП-5213.045430.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____ Ю.О. Олійник

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ А.О. Олійник

Київ – 2019 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмне забезпечення порівняння електронних та сканованих документів

Галузь застосування: робота користувача з документами різних форматів на комп’ютері

Наведене технічне завдання поширюється на розробку програмного забезпечення порівняння електронних та сканованих документів КПІ.ІП-5213.045430, котра використовується для порівняння на відповідність документів паперового та електронного формату та аналізу їх схожості та призначена для користувачів, що працюють з паперовим та електронним документообігом одночасно та повинні порівнювати достовірність інформації на обох типах відповідних документів..

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмного забезпечення порівняння електронних та сканованих документів є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для розпізнавання тексту в сканованих або сфотографованих документах та порівняння їх з електронною копією і виводу відповідного результату.

Метою розробки є підвищення ефективності роботи користувача при перевірці на відповідність документів в паперовому та електронному вигляді та зменшення ймовірності виникнення помилки при аналізі та висновка за допомогою надання зручного інтерфейсу, через який можна завантажити потрібні документи та отримати інформацію щодо їх схожості.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

- завантажувати вибрані файли;
- виводи тексти вибраних файлів;
- порівнювати вибрані файли;
- виводити процент схожості;
- виводити результати невідповідності.

4.1.2 Розробку виконати на платформі Windows 2007 та вище.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації документів.

4.3 Умови експлуатації

Не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

4.4.2 Наявність вільної пам’яті для встановлення програми та збереження результатів.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Windows 7 або вище.

4.5.2 Вхідні дані повинні бути представлені в наступному форматі: png, jpg, jpeg, doc, docx, dot, dotx, rtf, pdf, txt.

4.5.3 Результати повинні бути представлені в наступному форматі: дійсне число у вигляді проценту співпадіння та тексти з виділеними місцями невідповідності.

4.5.4 Програмне забезпечення повинно працювати з програмним забезпеченням Microsoft Word 2003 та вище.

4.5.5 Програмне забезпечення повинно працювати з програмним забезпеченням Adobe Reader 5.1 та вище.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 50 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Програма та методика тестування

5.4 Графічна частина повинна бути виконана на аркуші формату А3 та А4, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Схема структурна діяльності процесу роботи програмного забезпечення.

5.4.2 Схема бізнес-процесу роботи програмного забезпечення.

5.4.3 Схема структурна класів програмного забезпечення

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення рекомендованої літератури	22.02.2019	
2.	Аналіз існуючих методів розв’язання задачі	22.02.2019	
3.	Постановка та формалізація задачі	1.03.2019	Специфікації програмного забезпечення
4.	Аналіз вимог до програмного забезпечення	5.03.2019	Схема матриці трасування
5.	Алгоритмізація задачі	15.03.2019	Схема структурна діяльності роботи програмного забезпечення
6.	Моделювання програмного забезпечення	20.03.2019	Схема бізнес-процесу роботи програмного забезпечення
7.	Обґрунтування використовуваних технічних засобів	25.03.2019	
8.	Розробка архітектури програмного забезпечення	05.04.2019	Схема архітектурних компонентів

9.	Розробка програмного забезпечення	10.04.2019	Технічна документація
10.	Налагодження програми	15.05.2019	Програма та методика тестування
11.	Виконання графічних документів	20.05.2019	Графічний матеріал проекту
12.	Оформлення пояснювальної записки	20.05.2019	Пояснювальна записка

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до документа “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОРІВНЯННЯ ЕЛЕКТРОННИХ ТА СКАНОВАНИХ ДОКУМЕНТІВ

Програма та методика тестування

КПІ.ІП-5213.045430.04.51

“ПОГОДЖЕНО”

Керівник проекту:

_____ Ю.О. Олійник

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____А.О. Олійник

Київ – 2019 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробувань є програмне забезпечення, що порівнює електронні та скановані документи. Продукт створений на платформі Microsoft .Net з використанням технологій Windows Form, Tesseract.

2 МЕТА ТЕСТУВАННЯ

Метою тестування знаходження та виправлення помилок до того часу, як користувач почне працювати з програмою.

Під час тестування має бути перевірено наступне:

– функціональна робота програмного забезпечення;
– розрахунок, виведення та збереження результатів;
– розподіл тексту на блоки та збереження його в форматі JSON;
– запобігання некоректних дій користувача;
– зручність роботи з програмою;
– відповідність дизайну вимогам Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам.

Використовуються наступні методи:

– функціональне тестування;

– інтеграційне тестування;

– тестування продуктивності програмного забезпечення, зокрема Load testing (навантажувальне тестування);

– тестування інтерфейсу.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність програмного забезпечення провожиться способами:

– статичного тестування коду;

– динамічного ручного тестування на відповідність роботи програми при некоректних дій користувача;

– динамічного ручного тестування на відповідність функціональним вимогам;

– тестування програми з документами, створеними різними версіями програми Microsoft Word;

– тестування програми з документами, створеними різними версіями програми Adobe Reader;

– тестування при завантажені файлу великого розміру;

– тестування зручності використання;

– тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОРІВНЯННЯ ЕЛЕКТРОННИХ ТА СКАНОВАНИХ ДОКУМЕНТІВ

Керівництво користувача

КПІ.ІП-5213.045430.05.34

“ПОГОДЖЕНО”

Керівник проекту:

_____ Ю.О. Олійник

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____А.О. Олійник

Київ – 2019 року

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Система порівняння документів використовується для співставлення даних з файлів певних форматів та отримання інформації щодо відповідності. Програмне забезпечення містить простий та зрозумілий інтерфейс.

Функціональними можливостями програми є:

- завантаження документів;
- розпізнавання текстів та розподіл даних на блоки;
- порівняння даних та вивід показника схожості відповідно по блокам;
- виділення невідповідних символів;
- збереження результату порівняння в файл.

2 ПІДГОТОВКА ДО РОБОТИ

2.1 Системні та апаратні вимоги для використання продукту

Програмне забезпечення створено для використання на комп’бері або ноутбуці.

Системні вимоги:

- операційна система Windows 7 або наступний версій;
- наявність програми Microsoft Word версії 2007 або наступних;
- наявність програми Adobe Acrobat Reader DC версії 5.1 або наступних.

Апаратні вимоги:

- достатньо вільної пам’яті комп’ютері для встановлення програми;
- 32 або 64-розрядний процесор з тактовою частотою 1GHz;
- оперативна пам’ять не менше 1Gb.

2.2 Встановлення програмного забезпечення

Щоб почати роботу з програмою, необхідно запустити файл Comparator.exe. Якщо після запуску виведеться форма програмного забезпечення, то продукт встановлений успішно та працює.

3 РОБОТА З ПРОГРАМОЮ

Після успішно встановлення програмного забезпечення на екран виведеться форма, що зображеня на рисунку 1.1.

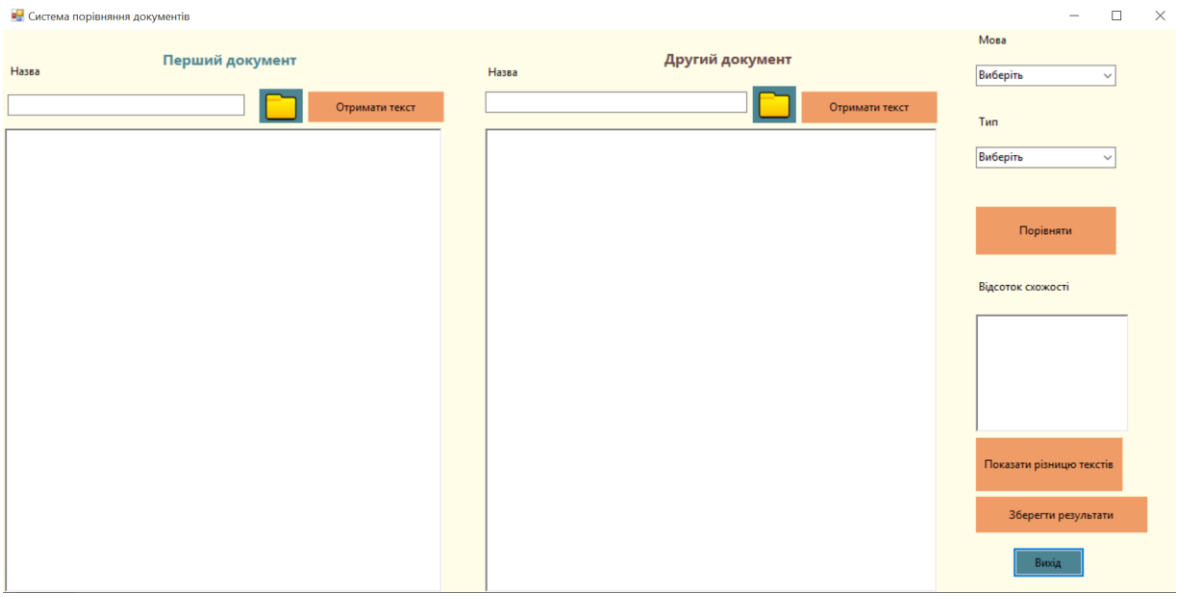
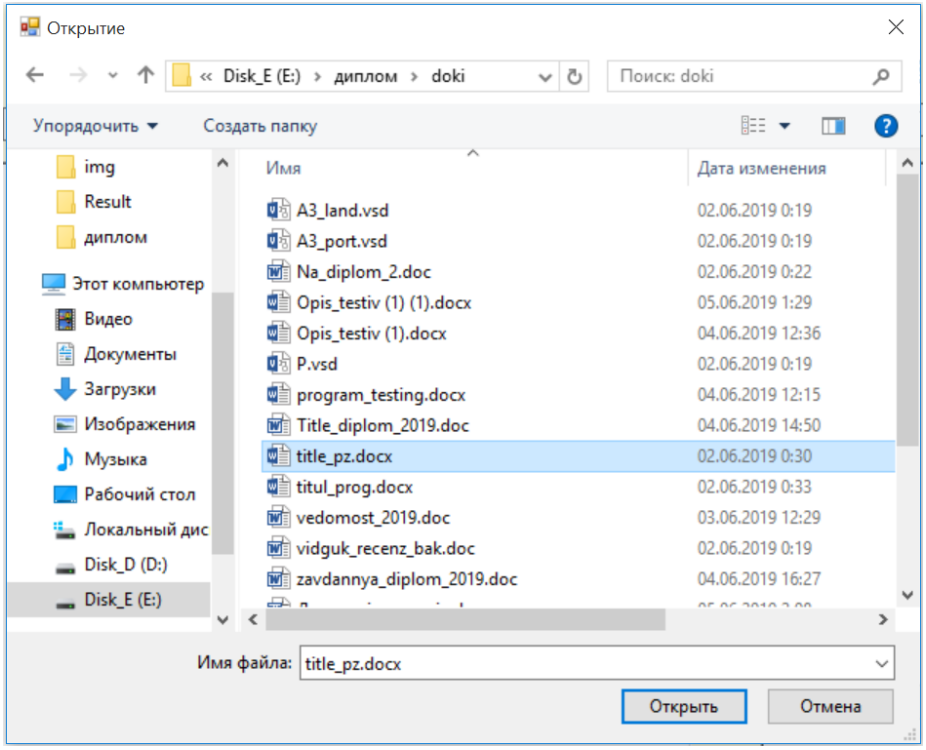


Рисунок 1.1 – Початкова форма програми

Щоб завантажити документи, потрібно в текстовому полі під словом «Назва» власноруч написати повністю шлях та назву, або натиснути кнопку із зображення папки. Після натискання виведеться вікно з файловою системою комп’ютера, що зображене на рисунку 1.2. Необхідно знайти та вибрати потрібний файл, натиснути кнопку «Відкрити».



Рисунко 1.2 – Вікно з файловою системою

Якщо користувач не вкаже повне ім’я файлу, то виведеться повідомлення, що зображено на рисунку 1.3.

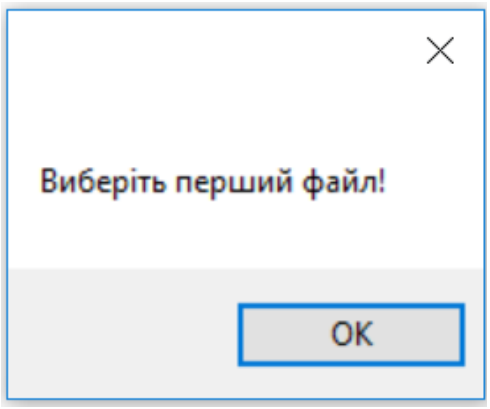


Рисунок 1.3 – Повідомлення про не вибраний відповідний файл

Після того, як вказані обидва файли, необхідно вибрати мову, на якій написані дані, та тип документів. Ці властивості представлені у вигляді випадаючого списку. Якщо користувач не вибрав мову або тип, то виведеться відповідне повідомлення, що забражено на рисунках 1.4 і 1.5.

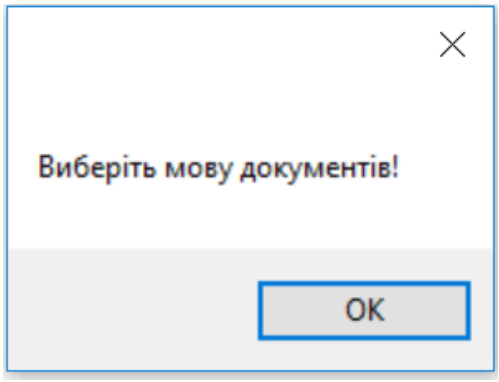


Рисунок 1.4 – Повідомлення про не вибрану мову

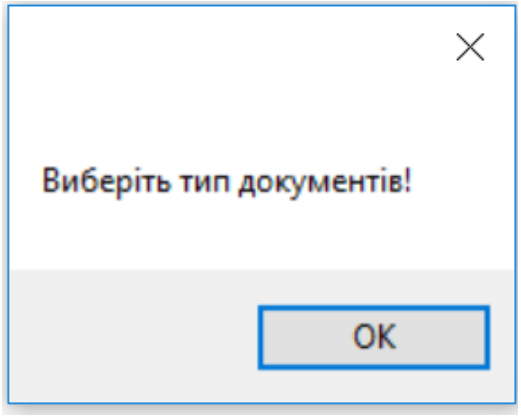


Рисунок 1.5 – Повідомлення про не вибраний тип

Після того, як користувач вибрав обидва документи, мову та тип, потрібно натиснути на обидві кнопки «Отимати текст», щоб виконалось розпізнавання та зчитування інормації з документів. Після завершення виконання функцій тексти виведяться у відповідні текстові поля, як показанона рисунку 1.6.

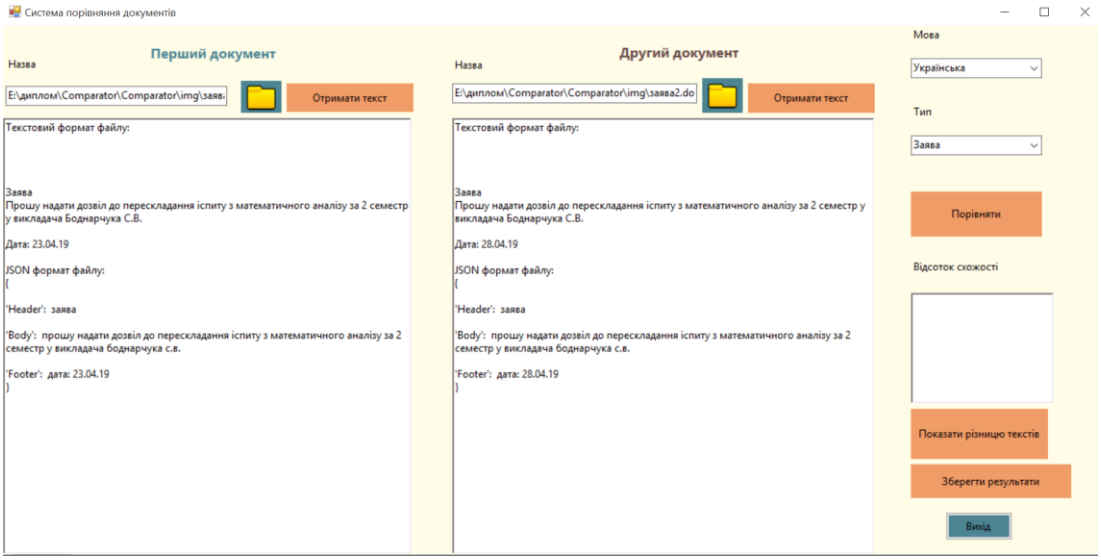


Рисунок 1.6 – Виведення результатів зчитування даних

Якщо користувач не обрав документи та не зчитав текст і натиснув кнопку «Порівняти», то виведеться повідомлення, як зображено на рисунку 1.7.

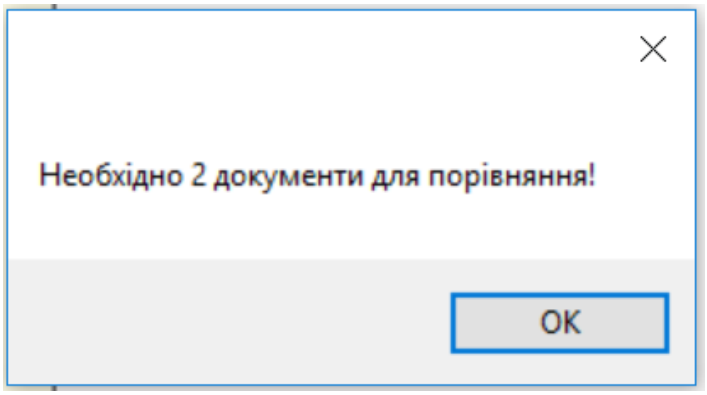


Рисунок 1.7 – Повідомлення, що не вибрані документи для порівняння

Щоб порівняти виведені тексти, користувач натискає кнопку «Порівняти» та виконується функція розрахування показника схожості. Результат видається у вигляді дійсного числа відсотка схожості по кожному блоці, як показано на рисунку 1.8.

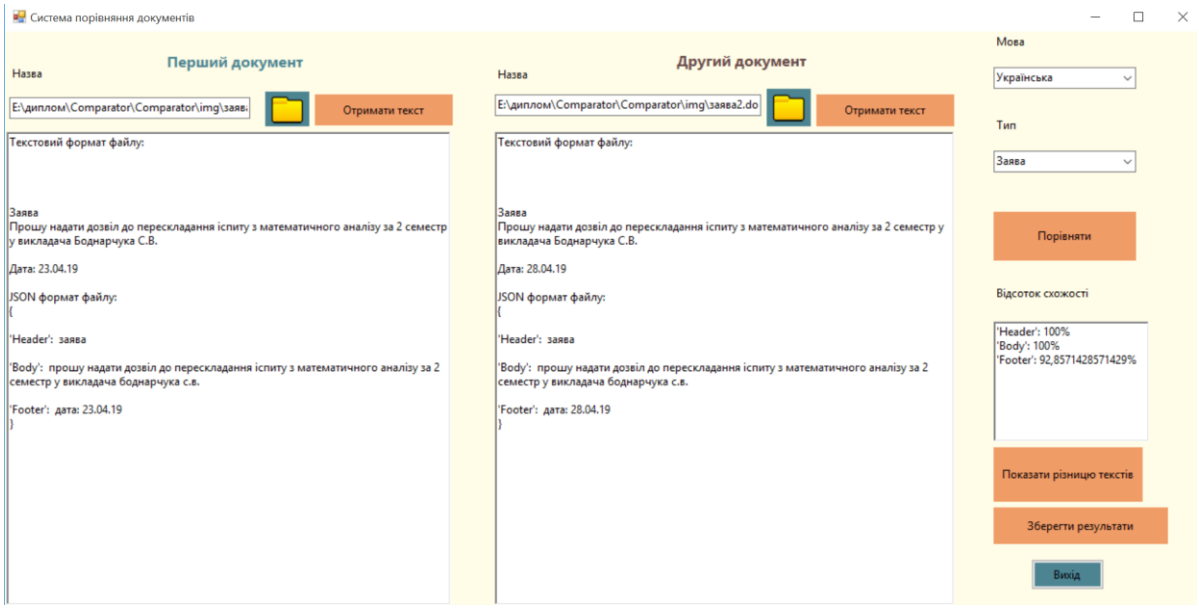


Рисунок 1.8 – Виведення результату показника схожості

Користувач натискає кнопку «Показати різницю текстів» та отримує тексти документів з виділеними частинами, що не співпадають у відповідних текстових полях, як зображено на рисунку 1.9.

Рисунок 1.9 – Виведення даних з виділеними символами, що не співпадають

Якщо користувач не вибрав документи та не зчитав текст, то виведеться повідомлення, як показано на рисунку 1.7.

Щоб виконати функцію збереження інформації в файл, користувач натискає кнопку «Зберегти результати». Після цього інформація з усіх текстових полів зберігається в файлі формату rtf в папці Result. При успішному завершені запису виведеться повідомлення, як зображено на рисунку 1.10.

Рисунок 1.10 – Повідомлення про успішний запису результату в файл

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОРІВНЯННЯ ЕЛЕКТРОННИХ ТА СКАНОВАНИХ ДОКУМЕНТІВ

Опис програми

КПІ.ІП-5213.045430.06.13

“ПОГОДЖЕНО”

Керівник проекту:

_____ Ю.О. Олійник

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____А.О. Олійник

Київ – 2019 року

Тексти програмного коду

Система порівняння документів

(Найменування програми (документа))

DVD-R

(Вид носія даних)

10 арк, 68,1 Мб

(Обсяг програми (документа) , арк.,) Мб)

Київ - 2019

```
namespace Comparator.FunctionLibrary
{
    public interface IFileBase    //інтерфейс, що має властивість назву документа
    {
        string NameOfFile { get; set; }

    }
}

namespace Comparator.FunctionLibrary
{
    public interface IResultBase      //інтерфейс, що має властивість результат фоункції  у вигляді тексту
    {
        string Result { get; set; }
    }
}

public class FileManager : IFileBase, IResultBase  //клас, отримує назву файлу, мову, та в залежності від вормату  виклакає відповідний клас для зчитування
{
    private string nameOfDoc;
    private string inforInDoc;
    private string language;
    private int fileClass;

    public FileManager()
    {
        NameOfFile = "";
        Language = "";
        Result = "";
    }
    public string Language
    {
        set { language = value; }

        get { return language; }
    }

    public string NameOfFile
    {
        set { nameOfDoc = value; }

        get { return nameOfDoc; }
    }

    public string Result
    {
        set { inforInDoc = value; }

        get { return inforInDoc; }
    }

    private string GetExtension()
    {
        string extension = Path.GetExtension(nameOfDoc);
        return extension;
    }

    private FileType GetFileType()
    {
```

```
        var type = FileType.Invalid;
        string extension = GetExtension();
        switch (extension)
        {
            case ".txt" : type = FileType.TxtType;
                break;
            case ".pdf" : type = FileType.PdfType;
                break;
            case ".doc" : type = FileType.DocType;
                break;
            case ".docx" : type = FileType.DocxType;
                break;
            case ".rtf" : type = FileType.RtfType;
                break;
            case ".dot" : type = FileType.DotType;
                break;
            case ".dotx" : type = FileType.DotType;
                break;
            case ".png" : type = FileType.GraphicType;
                break;
            case ".jpg" : type = FileType.GraphicType;
                break;
            case ".jpeg" : type = FileType.GraphicType;
                break;
            case ".tiff" : type = FileType.GraphicType;
                break;
        }

        return type;
    }

    public void ManageFile()
    {
        FileType t = GetFileType();
        if (t == FileType.GraphicType)
        {
            ReadImgFile img = new ReadImgFile();
            img.Language = Language;
            img.NameOfFile = NameOfFile;
            img.Result = Result;
            img.Read();
            Result = img.Result;
        }
        else
        {
            if (t == FileType.TxtType)
            {
                ReadTextClass text = new ReadTextClass();
                text.NameOfFile = NameOfFile;
                text.Result = Result;
                text.Read();
                Result = text.Result;
            }
            else
            {
                if (t == FileType.PdfType)
                {
                    ReadPdfFile pdf = new ReadPdfFile();
                    pdf.NameOfFile = NameOfFile;
                    pdf.Language = Language;
                    pdf.Result = Result;
                    pdf.Read();
```

```
                    Result = pdf.Result;
                }

                else
                {
                    ReadWordFile word = new ReadWordFile();
                    word.NameOfFile = NameOfFile;
                    word.Result = Result;
                    word.Read();
                    Result = word.Result;
                }

            }
        }
    }

public class ReadImgFile : FileManager  //клас, що розпізнає текст з зображеня
{
    private Bitmap img;
    private string language;

    public string Language
    {
        set { language = value; }

        get { return language; }
    }

    public void Read()
    {
        img = new Bitmap(NameOfFile);
        TesseractEngine engine = new TesseractEngine("./tessdata", language,
EngineMode.Default);
        Page page = engine.Process(img, PageSegMode.Auto);
        Result = page.GetText();
    }
}
public class ReadTextClass : FileManager   //клас, що зчитує текст з файлів
формату txt
{
    public new void Read()
    {

        Result = "";
        FileStream fstream = null;
        try
        {
            fstream = File.OpenRead(NameOfFile);

            Result = File.ReadAllText(NameOfFile);
        }
        catch (FileNotFoundException)
        {
            throw new FileNotFoundException();
        }
        catch (DirectoryNotFoundException)
        {
            throw new DirectoryNotFoundException();
        }
        catch (IOException exception)
        {
            throw new IOException(exception.Message);
```

```
        }
        finally
        {
            if (fstream != null)
                fstream.Close();
        }
    }
}
public class ReadWordFile : FileManager   //клас, що зчитує текс з файлів
форматів doc, docx, dot, dotx, rtf
{
    public ReadWordFile()
    {
        this.NameOfFile =
        this.Result = Result;
    }
    public void Read()
    {
        //Result = "";
        Application application = new Application();
        //object path = NameOfFile;
        object nullobj = System.Reflection.Missing.Value;
        try
        {
            Document document = application.Documents.Open(NameOfFile, ref
nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj,
                ref nullobj);
            int count = document.Words.Count;
            for (int i = 0; i < document.Paragraphs.Count; i++)
            {
                Result += document.Paragraphs[i + 1].Range.Text.ToString();
            }

            var doc_close = (_Document) document;
            doc_close.Close();
        }
        catch (FileNotFoundException)
        {
            throw new FileNotFoundException();
        }
        catch (DirectoryNotFoundException)
        {
            throw new DirectoryNotFoundException();
        }
        catch (IOException exception)
        {
            throw new IOException(exception.Message);
        }
        finally
```

```
            {
                application.Quit();
            }
        }

    }
    public class ReadPdfFile : FileManager  //клас, що зчитує текст із зображень від
    pdf файлу
    {
        private Bitmap img;
        private string language;

        public string Language
        {
            set { language = value; }

            get { return language; }
        }

        public void Read()
        {
            PdfToImg image = new PdfToImg(NameOfFile);
            image.ExtractTextPdf();
            List<string> pdfFiles = image.GetNameOfFIles();
            TesseractEngine engine = new TesseractEngine("./tessdata", language,
    EngineMode.Default);

            foreach (string nameOfImg in pdfFiles)
            {
                img = new Bitmap(nameOfImg);
                Page page = engine.Process(img, PageSegMode.Auto);
                Result += page.GetText();
                page.Dispose();
            }

        }
    }
    public class PdfToImg  //клас, що перевторює pdf файл в список зображень формату
    tiff
    {
        private int countOfPages;
        private string nameOfPdf;
        private List<string> outputFiles;

        public PdfToImg(string name)
        {
            nameOfPdf = name;
            outputFiles = new List<string>();
        }

        public void ExtractTextPdf()
        {
            string nameOfImg = @"E:\диплом\Comparator\PdfToImg\" + GetName() + "%d"
    + ".tiff";

            GhostscriptWrapper.GeneratePageThumbs(nameOfPdf, nameOfImg,1,
    GetCountOfPages(),210,297);
            for (int i = 1; i <= GetCountOfPages(); i++)
            {
                nameOfImg = @"E:\диплом\Comparator\PdfToImg\" + GetName() + i +
    ".tiff";
                outputFiles.Add(nameOfImg);
```

```
            }
        }

        private string GetName()
        {
            return Path.GetFileNameWithoutExtension(nameOfPdf);
        }
        private int GetCountOfPages()
        {

            StreamReader r = new StreamReader(File.OpenRead(nameOfPdf));Regex regex
= new Regex(@"/Type\s*/Page[^s]");
            MatchCollection matches = regex.Matches(r.ReadToEnd());

            countOfPages = matches.Count;
            return countOfPages;
        }

        public List<string> GetNameOfFIles()
        {
            return outputFiles;
        }
}
public class ComparisonTwoText : IResultBase   //клас, що вираховує показник
схожості за алгоритмом відстану Дамерау-Левенштейна
{
        private string resultOfComparing;
        private double countSimilarity;
        private double countDifference;
        private string lines1;
        private string lines2;

        public string Result
        {
            set { resultOfComparing = value; }

            get { return resultOfComparing; }
        }

        private void SplitString(string text1, string text2)
        {
            lines1 = Regex.Replace(text1, @"^\s+$[\r\n]*", string.Empty,
RegexOptions.Multiline).Trim();
            lines2 = Regex.Replace(text2, @"^\s+$[\r\n]*", string.Empty,
RegexOptions.Multiline).Trim();

            lines1 = Regex.Replace(lines1, @" {2,}", " ").ToLower();
            lines2 = Regex.Replace(lines2, @" {2,}", " ").ToLower();
        }

        private static int Min(int a, int b, int c, int d)
            => Math.Min(a, Math.Min(b, Math.Min(c, d)));

        public double ComparingText(string text1, string text2)
        {
            SplitString(text1, text2);
            if (lines1 == null)
            {
                throw new ArgumentNullException(nameof(lines1));
            }

            if (lines2 == null)
```

```
{
    throw new ArgumentNullException(nameof(lines2));
}

if (lines1.Equals(lines2))
{
    return 1.0;
}

int inf = lines1.Length + lines2.Length;
var da = new Dictionary<char, int>();

for (int d = 0; d < lines1.Length; d++)
{
    da[lines1[d]] = 0;
}

for (int d = 0; d < lines2.Length; d++)
{
    da[lines2[d]] = 0;
}
int[,] h = new int[lines1.Length + 2, lines2.Length + 2];
for (int i = 0; i <= lines1.Length; i++)
{
    h[i + 1, 0] = inf;
    h[i + 1, 1] = i;
}

for (int j = 0; j <= lines2.Length; j++)
{
    h[0, j + 1] = inf;
    h[1, j + 1] = j;
}
for (int i = 1; i <= lines1.Length; i++)
{
    int db = 0;
    for (int j = 1; j <= lines2.Length; j++)
    {
        int i1 = da[lines2[j - 1]];
        int j1 = db;

        int cost = 1;
        if (lines1[i - 1] == lines2[j - 1])
        {
            cost = 0;
            db = j;
        }

        h[i + 1, j + 1] = Min(
            h[i, j] + cost,  // Substitution
            h[i + 1, j] + 1, // Insertion
            h[i, j + 1] + 1, // Deletion
            h[i1, j1] + (i - i1 - 1) + 1 + (j - j1 - 1)
        );
    }

    da[lines1[i - 1]] = i;
}


double stepsToSame = h[lines1.Length + 1, lines2.Length + 1];;
double percentage = 1.0 - ((double)stepsToSame / (double)
```

```
Math.Max(lines1.Length, lines2.Length));

        return percentage;
    }
}
public class FileBlock   //клас, що описує формат JSON
{
    public string Header { get; set; }

    public string Body { get; set; }

    public string Footer { get; set; }
}
public class ConvertJS    //клас, що серилізує та десерилізує текст в формат
JSON
{
    private string textResult;
    private FileBlock file;

    public ConvertJS() {}

    public ConvertJS(FileBlock file)
    {
        this.file = file;
    }

    public FileBlock GetFileBlocks()
    {
        return file;
    }

    public string ConvertToJS()
    {
        textResult = JsonConvert.SerializeObject(file);
        return textResult;
    }

    public string ConvertFromJS()
    {
        FileBlock fileNew =
JsonConvert.DeserializeObject<FileBlock>(textResult);

        string result = "{\n\n'Header':  " + fileNew.Header +"\n\n'Body':  "+
fileNew.Body + "\n\n'Footer':  "+ fileNew.Footer + "\n}";
        return result;
    }
}
public class SaveResult     //клас, що зберігає результат в файл
{
    private string nameOfFile1;
    private string nameOfFile2;

    public SaveResult(string name1, string name2)
    {
        nameOfFile1 = name1;
        nameOfFile2 = name2;
    }

    private string GetNameOfFile()
    {
        string nameResult = Path.GetFileNameWithoutExtension(nameOfFile1) + "
порівняння " +
```

```
                                Path.GetFileNameWithoutExtension(nameOfFile2);
            return nameResult;
        }

        public void WriteResult(RichTextBox box1, RichTextBox box2, RichTextBox
box3)
        {
            RichTextBox temp = new RichTextBox();
            box1.SelectAll();
            box1.Copy();

            temp.Paste();
            box2.SelectAll();
            box2.Copy();
            temp.Paste();

            box3.SelectAll();
            box3.Copy();
            temp.Paste();

            temp.SaveFile(@"E:\диплом\Comparator\Result\" + GetNameOfFile() +
".rtf", RichTextBoxStreamType.RichText);
        }
    }
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОРІВНЯННЯ ЕЛЕКТРОННИХ ТА СКАНОВАНИХ ДОКУМЕНТІВ

Графічні матеріали

КПІ.ІП-5213.045430.07.99

“ПОГОДЖЕНО”

Керівник проекту:

_____ Ю.О. Олійник

Нормоконтроль:

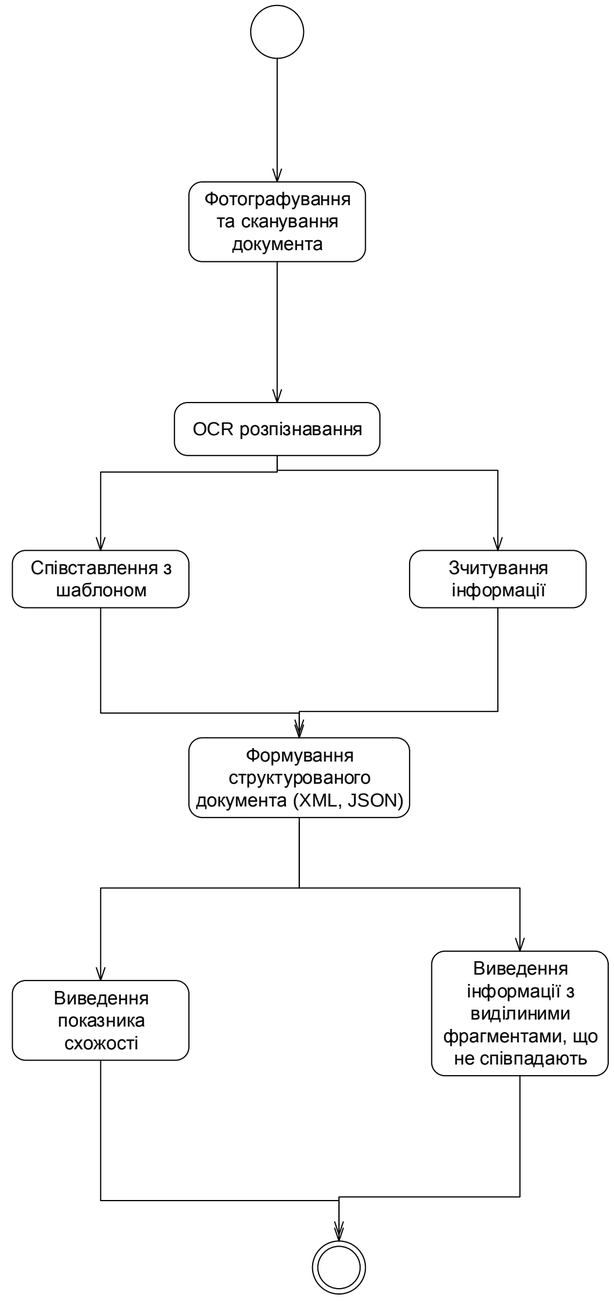
_____ К.І. Ліщук

Виконавець:

_____А.О. Олійник

Київ – 2019 року

КПІ.ІП-5213.045430.07.99.СС



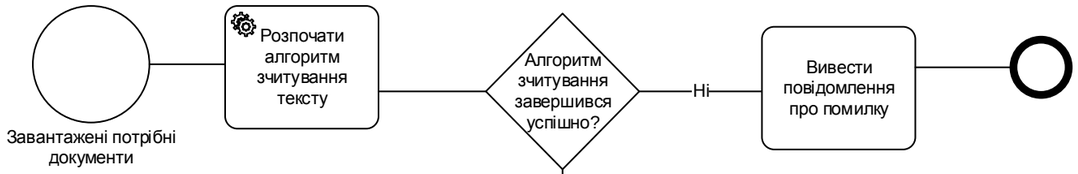
Інв. № підл.	Підп. дата	Інв. № взам. підл.	Інв. № дубл.	Підп. дата

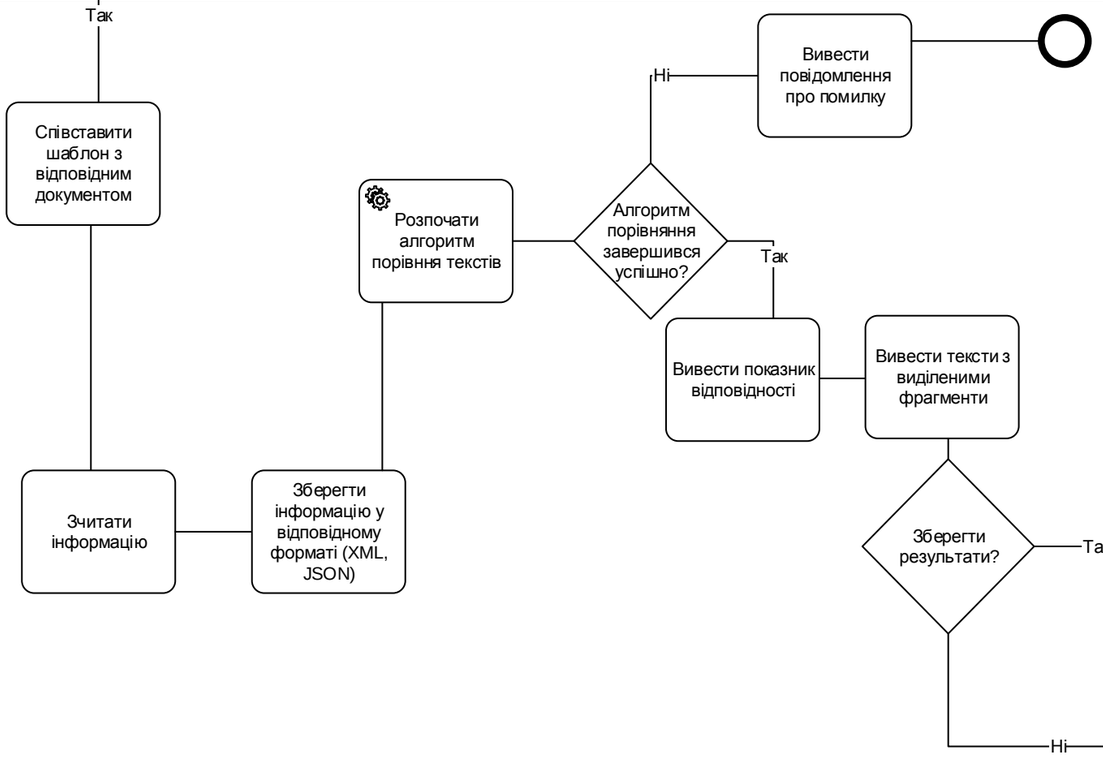
Зм.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-5213.045430.07.99.СС		
Розроб.		Олійник А.О.			Схема структурна діяльності	Лит.	Арк. 1 / Аркушів
Перев.		Олійник Ю.О.				Аркуш	Аркушів
Т. Кон.					Програмне забезпечення порівняння електронних та сканованих документів	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-52	
Н. Кон.		Ліщук К.І.					
Затв.		Олійник Ю.О.					

Система порівняння документів

Зчитування інформації

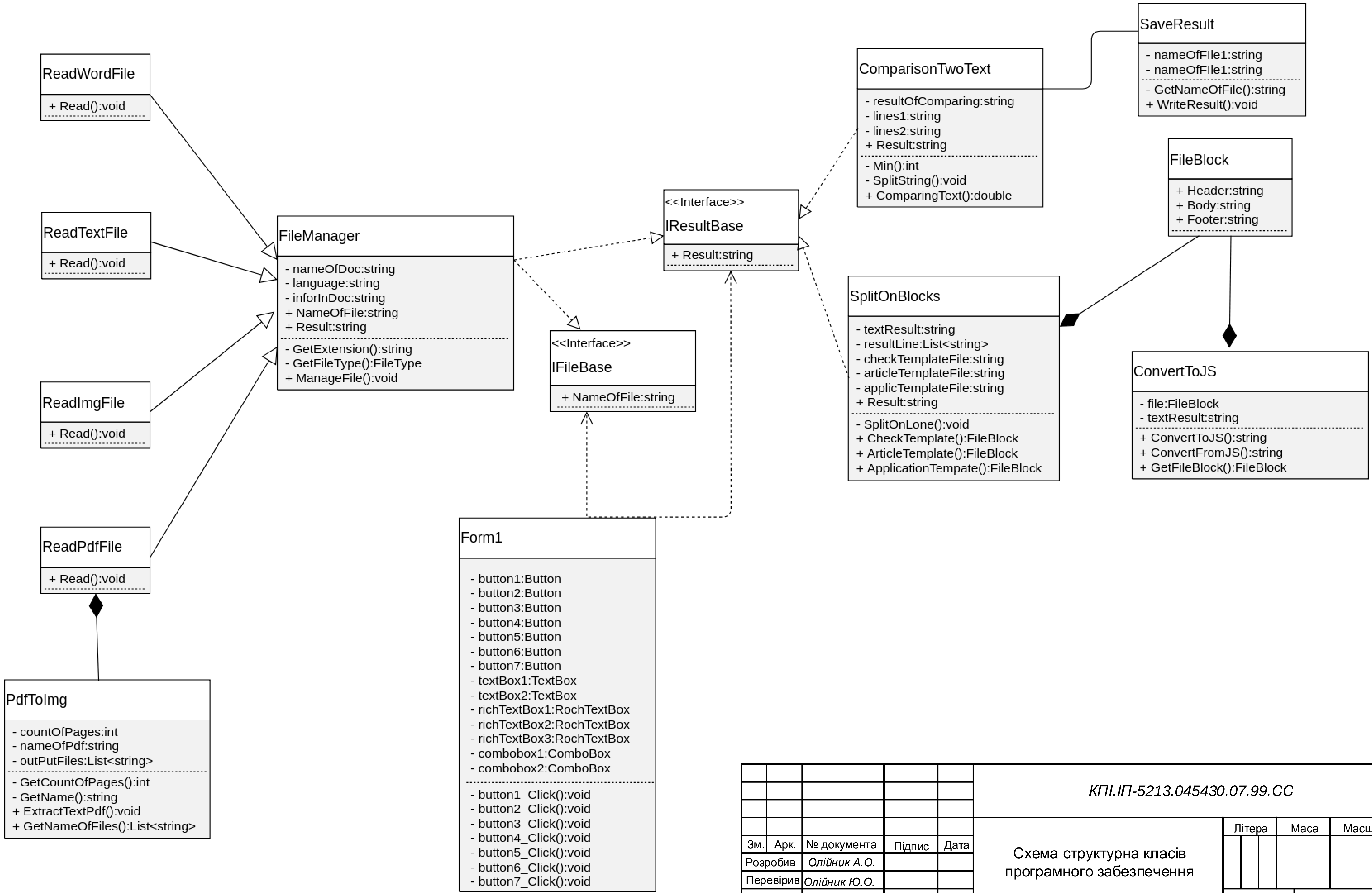
Порівняння даних





					КПІ.ІП-5213.045430.07.99.СБП			
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема бізнес-процесів			
Розробив		*Олійник А.О.*				Аркуш	Аркушів	
Перевірив		*Олійник Ю.О.*						
Т. кон.					*Програмне забезпечення порівняння електронних та сканованих документів*	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-52		
Н. кон.		*Ліщук К.І.*						
Затвердив		*Олійник Ю.О.*						

КПІ.ІП-5213.045430.07.99.СС



					КПІ.ІП-5213.045430.07.99.СС		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера	Маса	Масштаб
Розробив		*Олійник А.О.*						
Перевірив		*Олійник Ю.О.*				Аркуш	Аркушів	
Т. кон.					*Програмне забезпечення порівняння електронних та сканованих документів*	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-52		
Н. кон.		*Ліщук К.І.*						
Затвердив		*Олійник Ю.О.*						